

The University of Maine
DigitalCommons@UMaine

Electronic Theses and Dissertations

Fogler Library

12-1998

Topological Equivalence and Similarity in Multi-Representation Geographic Databases

Joao Argemiro de Carvalho Paiva

Follow this and additional works at: <http://digitalcommons.library.umaine.edu/etd>



Part of the [Geographic Information Sciences Commons](#)

Recommended Citation

Paiva, Joao Argemiro de Carvalho, "Topological Equivalence and Similarity in Multi-Representation Geographic Databases" (1998).
Electronic Theses and Dissertations. 593.
<http://digitalcommons.library.umaine.edu/etd/593>

This Open-Access Dissertation is brought to you for free and open access by DigitalCommons@UMaine. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of DigitalCommons@UMaine.

TOPOLOGICAL EQUIVALENCE AND SIMILARITY IN MULTI-REPRESENTATION GEOGRAPHIC DATABASES

By

João Argemiro de Carvalho Paiva

B.E. Federal University of Espírito Santo - Brazil, 1985

M.S. National Institute for Space Research - Brazil, 1994

A THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Spatial Information Science and Engineering)

The Graduate School

University of Maine

December 1998

Advisory Committee:

Max J. Egenhofer, Associate Professor of Spatial Information Science and Engineering,
Advisor

M. Kate Beard-Tisdale, Associate Professor of Spatial Information Science and
Engineering

Peggy Agouris, Assistant Professor of Spatial Information Science and Engineering

Robert D. Franzosa, Professor of Mathematics

David M. Mark, Professor of Geography, State University of New York, Buffalo

Library Rights Statement

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of Maine, I agree that the Library shall make it freely available for inspection. I further agree that permission for “fair use” copying of this thesis for scholarly purposes may be granted by the Librarian. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

João Argemiro de Carvalho Paiva

December 4, 1998

TOPOLOGICAL EQUIVALENCE AND SIMILARITY IN MULTI-REPRESENTATION GEOGRAPHIC DATABASES

By João A. C. Paiva

Thesis Advisor: Dr. Max J. Egenhofer

An Abstract of the Thesis Presented
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
(in Spatial Information Science and Engineering)

December, 1998

Geographic databases contain collections of spatial data representing the variety of views for the real world at a specific time. Depending on the resolution or scale of the spatial data, spatial objects may have different spatial dimensions, and they may be represented by point, linear, or polygonal features, or combination of them. The diversity of data that are collected over the same area, often from different sources, imposes a question of how to integrate and to keep them consistent in order to provide correct answers for spatial queries. This thesis is concerned with the development of a tool to check topological equivalence and similarity for spatial objects in multi-representation databases. The main question is what are the components of a model to identify topological consistency, based on a set of possible transitions for the different types of spatial representations. This work develops a new formalism to model consistently spatial objects and spatial relations between several objects, each represented at multiple levels of detail. It focuses on the topological consistency constraints that must hold among the different representation of objects, but it is not concerned about generalization operations of how to derive one representation level from another. The result of this thesis is a

computational tool to evaluate topological equivalence and similarity across multiple representations. This thesis proposes to organize a spatial scene –a set of spatial objects and their embeddings in space– directly as a relation-based model that uses a hierarchical graph representation. The focus of the relation-based model is on relevant object representations. Only the highest-dimensional object representations are explicitly stored, while their parts are not represented in the graph.

Acknowledgements

I gratefully acknowledge the guidance and support from the members of my advisory committee, Max Egenhofer, Kate Beard, Dr. Peggy Agouris, Robert Franzosa, and David Mark. In particular, I would like to thank my thesis advisor Max Egenhofer whose encouragement and advice were always present when needed. To all my friends at the Department of Spatial Information Science and Engineering I would like to thank you for making my study life a wonderful time. Some of these friends I would like to mention for having a close relationship in classes or developments of research topics: Andreas Blaser, John Florence, Doug Flewelling, Andrea Rodriguez, Kathleen Hornsby, Nectaria Tryfona, Roop Goyal, Dieter Pfoser, Michela Bertolloto, and Jayant Sharma. I would also like to thank Ubirajara Freitas, Virgínia Ragoni Correia, and Marisa da Motta, members of my Division at National Institute for Space Research (INPE - Brazil) who were part time at NCGIA in Maine.

The successful completion of this thesis would not have been possible without the support and love of my wife Marici Paiva, my sons Gustavo and Gabriel, and my daughter Camila.

Special thanks go to Brazilian Government and INPE for giving me four years to complete this program, and to Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) that granted a scholarship under number 200073/94-0. This work

was partially funded by the National Center for Geographic Information and Analysis under NSF grants SBR-8810917 and SBR-9700465, and by a Massive Digital Data Systems contract sponsored by the Advanced Research and Development Committee of the Community Management Staff (Principal Investigator: Max J. Egenhofer).

Table of Contents

| | |
|--|------|
| Acknowledgements | ii |
| List of Figures | x |
| List of Tables..... | xiii |
| Chapter 1: Introduction | 1 |
| 1.1 Multiple Representations | 3 |
| 1.2 Typical Changes through Multiple Representations | 5 |
| 1.3 Assessing Equivalence and Similarity | 6 |
| 1.4 Motivation..... | 9 |
| 1.5 Hypothesis..... | 10 |
| 1.6 Approach..... | 11 |
| 1.7 Results..... | 14 |
| 1.8 Intended Audience | 15 |
| 1.9 Thesis Organization | 15 |
| Chapter 2: Multiple Representations..... | 18 |
| 2.1 Database Design..... | 19 |
| 2.1.1 GEODYSSEY Database Design..... | 20 |
| 2.1.2 Links Between Representations..... | 21 |
| 2.2 Automated Generalization | 23 |
| 2.2.1 Spatial Knowledge for Generalization..... | 25 |

| | |
|--|----|
| 2.2.2 Model Generalization..... | 28 |
| 2.2.3 Cartographic Generalization | 31 |
| 2.2.3.1 Linear Feature | 34 |
| 2.2.3.2 Area feature | 35 |
| 2.2.3.3 Graph Approach..... | 37 |
| 2.3 Consistency Among Representations..... | 38 |
| 2.4 Summary | 40 |
| Chapter 3: Mathematical Background for Assessing Topological Consistency | 42 |
| 3.1 Topological Space..... | 43 |
| 3.2 Point-Set Topology | 43 |
| 3.2.1 Interior | 44 |
| 3.2.2 Closure | 44 |
| 3.2.3 Boundary..... | 45 |
| 3.2.4 Relationships between interior, closure, and boundary | 45 |
| 3.3 Cell Complexes | 45 |
| 3.4 Topological Homeomorphism | 46 |
| 3.5 Topological Relation Model | 47 |
| 3.5.1 Content Invariants..... | 48 |
| 3.5.2 Component Invariants..... | 50 |
| 3.6 Similarity Analysis..... | 52 |
| 3.7 Graph Model | 54 |
| 3.7.1 Isomorphism and Homeomorphism..... | 55 |
| 3.7.2 Subgraph | 56 |

| | |
|---|----|
| 3.8 Association graph for scene matching | 56 |
| 3.9 Summary | 59 |
| Chapter 4: A Qualitative Spatial Model for Multiple Representations | 60 |
| 4.1 Relation-Based Model..... | 61 |
| 4.1.1 Hierarchical Graph Representations | 63 |
| 4.1.2 Graph Modeling | 65 |
| 4.1.3 Adding Symbolic Geometric Information | 69 |
| 4.2 Equivalence Between Cell Complexes and the Relation-Based Model..... | 71 |
| 4.3 Mapping Cell Complexes into the Relation-Based Model | 72 |
| 4.3.1 Algorithm to Convert a Cell Complex into a Relation-Based Model..... | 73 |
| 4.3.2 Converting Individual Features | 74 |
| 4.3.3 Example | 75 |
| 4.4 Mapping the Relation-Based Model onto Cell Complexes | 79 |
| 4.4.1 Algorithm to Convert the Graph Representation onto the Cell Complex.. | 79 |
| 4.4.2 Converting Individual Nodes Representing Regions..... | 81 |
| 4.4.3 Example | 81 |
| 4.5 Summary | 82 |
| Chapter 5: Assessing Topological Equivalence | 83 |
| 5.1 Spatial Objects | 84 |
| 5.1.1 Simple Objects..... | 84 |
| 5.1.2 Relation Between Objects..... | 84 |
| 5.1.3 Complex Objects - Region with Holes | 84 |
| 5.1.3.1 Hole Characteristics..... | 85 |

| | |
|--|-----|
| 5.1.3.2 Characteristics of the Generalized Region..... | 87 |
| 5.1.3.3 Checking Equivalence | 88 |
| 5.2 Spatial Scenes | 89 |
| 5.2.1 Building Graph Structure and Association Graph | 90 |
| 5.2.2 Finding Isomorphic Configurations..... | 92 |
| 5.2.3 Validating Isomorphic Configurations | 95 |
| 5.2.4 General Procedure..... | 96 |
| 5.3 Similarity Measures | 98 |
| 5.3.1 Individual Representation..... | 99 |
| 5.3.1.1 Dimension..... | 99 |
| 5.3.1.2 Number of Adjacent Elements..... | 99 |
| 5.3.1.3 Adjacent Structure | 101 |
| 5.3.1.4 Hierarchical Structure..... | 101 |
| 5.3.2 Spatial Scenes | 102 |
| 5.3.2.1 Detailed Similarity..... | 102 |
| 5.3.2.2 Topological Similarity..... | 103 |
| 5.3.2.2.1 Valid Matching..... | 104 |
| 5.3.2.2.2 Feature Dimensions..... | 105 |
| 5.3.2.2.3 Spatial Relation between Features..... | 106 |
| 5.3.2.2.4 Graph Structure | 108 |
| 5.4 Topological Changes..... | 109 |
| 5.4.1 Merging..... | 109 |
| 5.4.2 Dropping | 114 |
| 5.5 Summary | 114 |

| | |
|--|-----|
| Chapter 6: Software Implementation | 116 |
| 6.1 UML Notation..... | 118 |
| 6.1.1 Class Diagrams | 118 |
| 6.1.2 Relationships..... | 119 |
| 6.2 Relation-Based Model Class Structure | 120 |
| 6.2.1 Feature | 123 |
| 6.2.2 Boundary..... | 124 |
| 6.2.3 Graph | 125 |
| 6.2.4 Spatial Scene..... | 126 |
| 6.3 Additional Classes..... | 127 |
| 6.3.1 Matching Pair..... | 128 |
| 6.3.2 Isomorphic Configuration..... | 129 |
| 6.3.3 Association Graph..... | 129 |
| 6.4 SPRING Model | 130 |
| 6.4.1 Converting SPRING's Vector Model into the Relation-Based Model.... | 133 |
| 6.4.2 Database Schema | 136 |
| 6.5 Summary | 139 |
| Chapter 7: Conclusions | 140 |
| 7.1 Summary | 140 |
| 7.2 Major Findings..... | 141 |
| 7.3 Future Work | 144 |
| 7.3.1 Integration of Spatial Relationships..... | 145 |
| 7.3.2 Extension to Linear Features | 146 |

| | |
|--|-----|
| 7.3.3 Integration with Model Generalization | 147 |
| 7.3.4 Interaction with Multi-Modal Languages | 148 |
| References | 150 |
| Appendix: Classes Specification | 166 |
| A.1 Feature | 166 |
| A.2 PointFeature | 169 |
| A.3 LinearFeature | 169 |
| A.4 AreaFeature | 170 |
| A.5 Boundary | 171 |
| A.6 Graph | 174 |
| A.7 SpatialScene | 177 |
| A.8 MatchingPair | 182 |
| A.9 IsomorphicConf | 183 |
| A.10 AssociationGraph | 185 |
| Biography of the Author | 188 |

List of Figures

| | |
|--|----|
| Figure 1.1: Multiple representation formats..... | 3 |
| Figure 1.2: Multiple representations for spatial objects..... | 3 |
| Figure 1.3: Multiple representation changes | 6 |
| Figure 1.4: Multiple representation framework. | 9 |
| Figure 2.1: GEODYSSEY multi-scale database design..... | 21 |
| Figure 2.2: Multiple topological views for Brazil: country, regions, states..... | 22 |
| Figure 2.3: Some generalization operations..... | 24 |
| Figure 2.4: Constraint-based framework (Beard 1991)..... | 26 |
| Figure 2.5: Framework for automated generalization (Brassel and Weibel, 1988). | 27 |
| Figure 2.6: Original line and its BLG-tree. | 35 |
| Figure 2.7: Acceptable topological transformations. | 36 |
| Figure 2.8: Homeomorphism concept..... | 40 |
| Figure 3.1: Point, line, and region: (a) interior; (b) closure; (c) set-theoretic boundary ... | 44 |
| Figure 3.2: Topological homeomorphism..... | 47 |
| Figure 3.3: Eight topological relations between two regions in \mathfrak{R}^2 | 49 |
| Figure 3.4: A <i>meet</i> relation with three boundary-boundary components..... | 51 |
| Figure 3.5: Component invariants of topological relations..... | 52 |
| Figure 3.6: Conceptual neighborhoods of topological relations between simple regions. | 54 |
| Figure 3.7: Graph and its adjacency matrix. | 55 |
| Figure 3.8: Homeomorphic and isomorphic graphs..... | 56 |

| | |
|---|-----|
| Figure 3.9: A graph and two of its subgraphs. | 56 |
| Figure 3.10: Two spatial scenes, their graphs, and the equivalent association graph. | 58 |
| Figure 4.1: A spatial scene and its equivalent graph. | 63 |
| Figure 4.2: Hierarchical levels of a spatial scene and their graph representations. | 64 |
| Figure 4.3: Spatial scene and its graph simplifications. | 66 |
| Figure 4.4: A spatial scene with a <i>cover</i> relation. | 69 |
| Figure 4.5: Dropping points from a region boundary. | 72 |
| Figure 4.6: Spatial scene: (a) cell features, and (b) graph representation. | 76 |
| Figure 5.1: Region with holes. | 87 |
| Figure 5.2: Hole H_o has a multi-meet. | 87 |
| Figure 5.3: Adjacent circles representing two spatial scenes. | 91 |
| Figure 5.4: Association graph between scenes of Figure 5.3. | 92 |
| Figure 5.5: Selected feature and its adjacent features. | 100 |
| Figure 5.6: Political States of Brazil. | 110 |
| Figure 5.7: Adjacent features for: (a) merged features; (b) individual feature. | 112 |
| Figure 6.1: Class icons with attributes and operations. | 118 |
| Figure 6.2: Types of UML relationships. | 120 |
| Figure 6.3: Class hierarchy for the relation-based model. | 122 |
| Figure 6.4: List and its node class structure. | 122 |
| Figure 6.5: Diagram for the <i>Feature</i> class and its child classes. | 124 |
| Figure 6.6: Diagram for <i>Boundary</i> class. | 125 |
| Figure 6.7: Diagram for <i>Graph</i> class. | 126 |
| Figure 6.8: Diagram for <i>SpatialScene</i> class. | 127 |
| Figure 6.9: Class diagrams of additional classes used for the topological checker. | 128 |

| | |
|---|-----|
| Figure 6.10: Diagram with attribute and operations for <i>MatchingPair</i> class..... | 129 |
| Figure 6.11: Diagram for <i>IsomorphicConf</i> class..... | 130 |
| Figure 6.12: Diagram for the <i>AssociationGraph</i> class..... | 130 |
| Figure 6.13: SPRING abstraction levels..... | 132 |
| Figure 6.14: SPRING's conceptual model..... | 133 |
| Figure 6.15: Representation model of SPRING..... | 134 |
| Figure 6.16: Vector data structure in SPRING..... | 136 |
| Figure 6.17: Modified representation model for SPRING..... | 137 |
| Figure 6.18: SPRING database schema to support the relation-based representation. ... | 138 |

List of Tables

| | |
|---|-----|
| Table 2.1: Topological generalization operators and object properties and relations | 30 |
| Table 2.2: Weibel's (1996) constraints for cartographic generalization. | 33 |
| Table 4.1: Relation-based model HG for Figure 4.3a. | 68 |
| Table 4.2: Complementary row to Table 4.1 that describes scene of Figure 4.4. | 69 |
| Table 4.3: Relation-based model HG^+ for Figure 4.3a. | 71 |
| Table 4.4: Equivalent graph nodes for 2-cells of Figure 4.7. | 77 |
| Table 4.5: HG^+ model components for Figure 4.6. | 78 |
| Table 5.1: Relation-based model information for scenes of Figure 5.3. | 91 |
| Table 5.2: Isomorphic configuration process based on Figure 5.4. | 95 |
| Table 5.3: Boundary components sequence for nodes of Table 5.2. | 96 |
| Table 5.4: Difference matrix for the conceptual neighbors between regions. | 107 |
| Table 5.5: Similarity values between features. | 111 |
| Table 5.6: Meet and adjacent structure similarity values between features. | 113 |

Chapter 1

Introduction

Geographic databases contain collections of spatial data representing a variety of views of the real world at a specific time. The term *spatial* refers to the location of objects positioned in geographic space. *Spatial objects* are representations of the elements of the real world such as rivers, countries, railways, and schools. Depending on the level of detail, such spatial objects may have different spatial dimensions, and they may be represented by point, linear, polygonal features, or combinations of these features. For example, the source of a river may be represented by a point feature, some channels of this river may be represented by linear features, and in parts where the river widens it may be represented by a polygonal feature. Each spatial object is described by spatial and non-spatial attributes. Spatial attributes, such as shape, area, length, perimeter, and volume, are usually derived from the positional and metric information of the object. Spatial objects share spatial relationships, which describe *topological* properties such as connectivity, orientation, adjacency, and containment.

Topological information is an essential component of any geographic database and geographic information systems (GISs). GISs are computational systems that deal with spatial objects. A GIS can be considered a toolbox that contains modules for acquisition, storage, maintenance, analysis, and display of spatial objects (Burrough 1986). Generally

geographic databases store topological information, which permits users to derive the spatial relationships between objects. Such topological information is usually stored at the representational level of the objects. For example, a line feature would include information about which other lines are connected with it and to which polygons it belongs. Topological representations have become the common method for organizing spatial objects in GISs. Topological models have been described by Corbett (1979), Frank and Kuhn (1986), (Kuijpers et al. 1995), and their implementational aspects by Egenhofer *et al.* (1989) and by Jackson (1989). Most current GISs only deal with a single model of the world. Spatial queries are based on the topological properties of the objects, and a single topological representation may take a long time to answer topological queries about large-scale objects. Because at a single level, a geographic database contains much more information than necessary to answer a spatial query. Advanced geographic databases have been proposed using the approach of multiple topological representations (Bruegger and Kuhn 1991) that allow access to topological information at different levels of detail.

The term multiple representations in GIS can be viewed in two different ways. The first one refers to the use of multiple spatial data models for the same data, and the second one refers to multiple geometric representations of individual spatial objects represented in the spatial data model (Camara et al. 1994). [Figure 1.1](#) shows the concept of multiple spatial data models for one type of information. For example, thematic data such as land use can be stored as a thematic image in the form of a regular raster or it can be recorded as vector data, containing the geometric features of the map. For data that correspond to a digital terrain model, such as the topography of a study area, the representation may vary between isolated samples, regular or irregular grids, and contour lines. [Figure 1.2](#)

illustrates the concept of multiple representations for spatial objects. The objects Amazonas and Xingú rivers have multiple geometric representations that may be stored in one or several map layers. This thesis addresses the problem of topological equivalence and similarity for spatial objects across these multiple geometric representations.

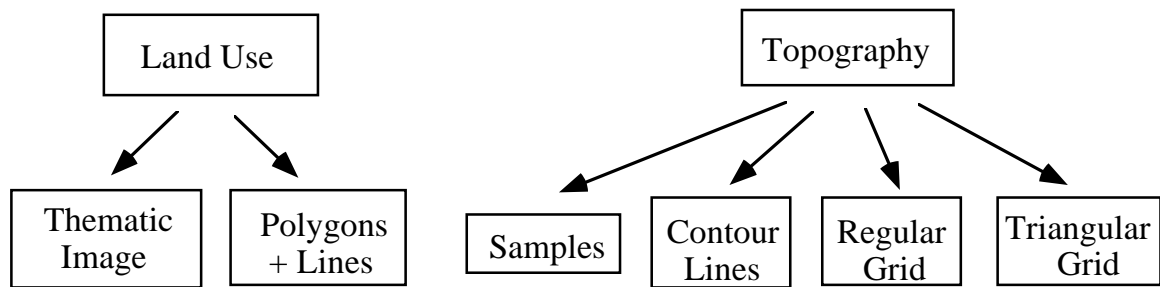


Figure 1.1: Multiple representation formats.

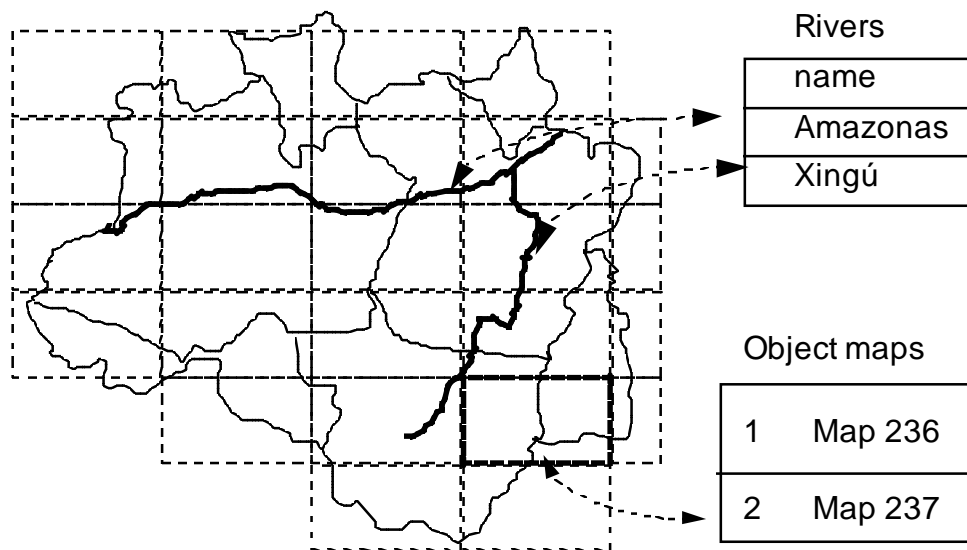


Figure 1.2: Multiple representations for spatial objects.

1.1 Multiple Representations

Multiple representations of geographic objects in spatial databases have started to emerge as a research topic in the geographic information community in the 1980s (Buttenfield

1989). The amount of spatial data available has grown considerably and they are available in different formats, in different scales, and they are usually generated by different procedures. Such multiple representations imply a considerable increase in the amount of data to be stored, introducing additional problems for the maintenance and integration of these data at different levels of detail. Progress in this field has been concentrated in three different areas: database issues, generalization, and spatial modeling (Buttenfield 1993). Database issues include concerns about organization of topological and metrical information for efficient access (Bruegger and Frank 1989); about linking and maintaining among consistently different representations of spatial data that are stored in a collection of maps for a specific area (Beard 1989); and about incorporating expert knowledge to produce spatial rules in order to preserve the database consistency (Mark 1991). Generalization issues are related to simplification of cartographic lines (Douglas and Peucker 1973) in general with regard to display objectives. Initial efforts in generalization have concentrated on geometric filtering and smoothing, but some new generalization algorithms or post-processing algorithms are trying to preserve the general structure of the data. Müller (1990), for instance, proposed a post-processing procedure to clean up self-intersections generated by line simplification algorithms lacking topologic control. Wang and Müller (1993) proposed a combination of procedural algorithms and predicate logic formalisms to generalize complex coastlines. Other approaches include the automated generalization of area patches over a two-dimensional space (Muller and Zeshen 1992) and a polygon mosaic simplification (de Berg et al. 1995), which considers objects from different classes and avoids self-intersections and the overlapping of neighboring lines. Some recent work about model generalization (Puppo and Dettori 1995; Tryfona and Egenhofer 1997), is more concentrated about topological properties

than metric information. Spatial modeling issues are concerned with the scale at which several geographic processes are likely to impact the structure of geographic features.

1.2 Typical Changes through Multiple Representations

Multiple representations of spatial data encompass changes in the geometric and topological structure of a geographic object. These changes may occur with the value of the resolution at which the object is encoded for computer storage, analysis, and depiction (Buttenfield 1989). The concept of multiple representation in GIS means that the geographic object may be represented in several different ways, each one to satisfy the needs of different users or analysis operations. Spatial representations at different scales can differ both in accuracy and resolution (Dettori and Puppo 1996). A less precise representation means that the data contain simplifications of the original representation, but the topology should not be changed. On the other hand, the reduction of the map resolution may change the topological structure of a spatial object, as well as the shape. Variations in resolution may affect the metric and topological aspects of a spatial representation. [Figure 1.3](#) shows some possible changes in a multiple representation environment. Metric changes are related to reduction in size and simplification of shape, while topological changes corresponds to the removal of small parts, merging of parts, and changes in the dimension representation of a spatial object. Shea and McMaster's (1991) cartographic generalization operators for simplification and smoothing are related to changes of the shape of the geographic feature, while operators for aggregation, amalgamation, merging, and collapsing are related to topological changes on the geographic feature.

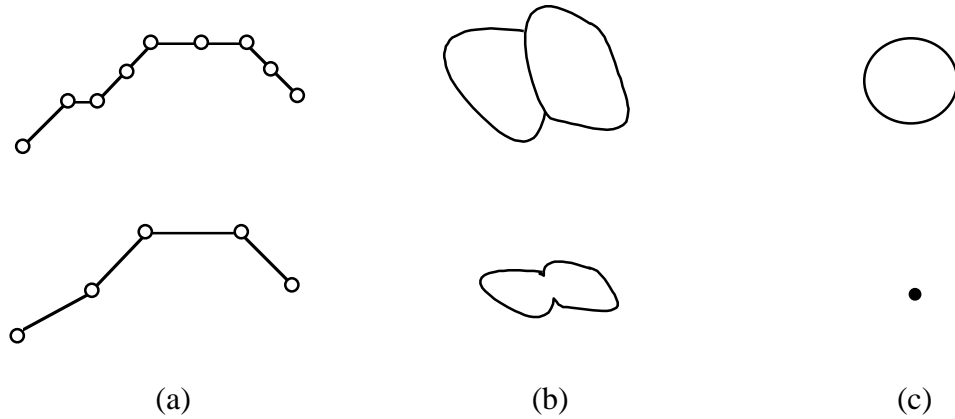


Figure 1.3: Multiple representation changes: (a) metric (simplification); (b) topological (merging); and (c) topological (collapsing).

1.3 Assessing Equivalence and Similarity

Ideally, multiple representation databases should be automatically derived from a single detailed representation in order to answer some specific user queries (Beard 1988). This automatic approach would avoid the problem of maintaining additional information for the same data. The National Spatial Data Infrastructure (NSDS 1993) makes this assumption as well. Unfortunately, this is not feasible at this time due to the inadequacy of automated generalization procedures. In general, the generalization algorithms are based only on the geometric part of linear elements, ignoring the fact that this linear feature may carry some topological structure, which should be preserved during simplification. The generalization procedures should take into consideration other constraint types such as metrical, topological, semantical, and gestalt (Weibel 1996). Müller *et al.* (1995) identified two conceptual levels for map generalization: cartographic generalization, which attempts to eliminate visual conflicts; and model generalization, which addresses reduction in detail at the representational level, relying on semantic abstraction mechanisms. Within the realm of model-based generalization, Tryfona and

Egenhofer (1997) developed a systematic model for the constraints that must hold with respect to spatial objects when two parts of an object are aggregated.

McMaster and Veregin (1996) described some approaches to provide multiple representation of a database:

- Creation of multi-scale versions for the same data by acquiring the information for different scales. Multi-scale databases have multiple representations for one object, each one for the respective scale.
- Development of robust data structures to support multiple representations, such as simplicial data structure (Jones et al. 1995).
- Application of generalization algorithms to create multiple versions of a database.

Multiple resolution databases have a close relationship with cartographic generalization. By applying generalization algorithms, new data can be generated, and it is important that the generalization procedure preserves the general structure of the data in order to avoid incorrect answers for queries performed at different levels of detail. We expect to maximize the database integrity between different levels of information for a multi-scale database. The creation of multiple representation data derived from automated generalization procedures requires a better knowledge of the mathematical and geometric behavior of these operations. It is necessary to develop computational tools that permit us to verify the overall quality of the generalization result in terms of topological, directional and semantical properties. [Figure 1.4](#) shows a framework for creating multiple representations of spatial data. Original data may be processed through different generalization algorithms, which generate different data for the same area. In order to test

the quality of these multiple data, a similarity checker may be applied to verify if the new data are equivalent or similar to the original one. Equivalence means that the resultant data preserve all the properties of the original data. Similarity is a deviation from equivalence. In this thesis 100% of similarity corresponds to the term equivalence. This similarity checker should include topological, directional, metrical, and semantical procedures. This thesis focuses on the topological properties. Equivalence between different levels of detail rarely occurs in practice after a simplification of the data, and a relaxation model may be applied in order to identify levels of similarity between the new data and the original one. In this thesis, the equivalence and similarity analysis are based on a relation-based model that represents a spatial scene as a hierarchy of graphs. This relation-based model is built from the topological relationships between the spatial object representations.

The term generalization can be associated with cartographic generalization, which is concerned with shape of the geometry, or it can be associated with model generalization, which is concerned with qualitative topologic information. Therefore, model generalization is more closely related with the qualitative model developed in this thesis.

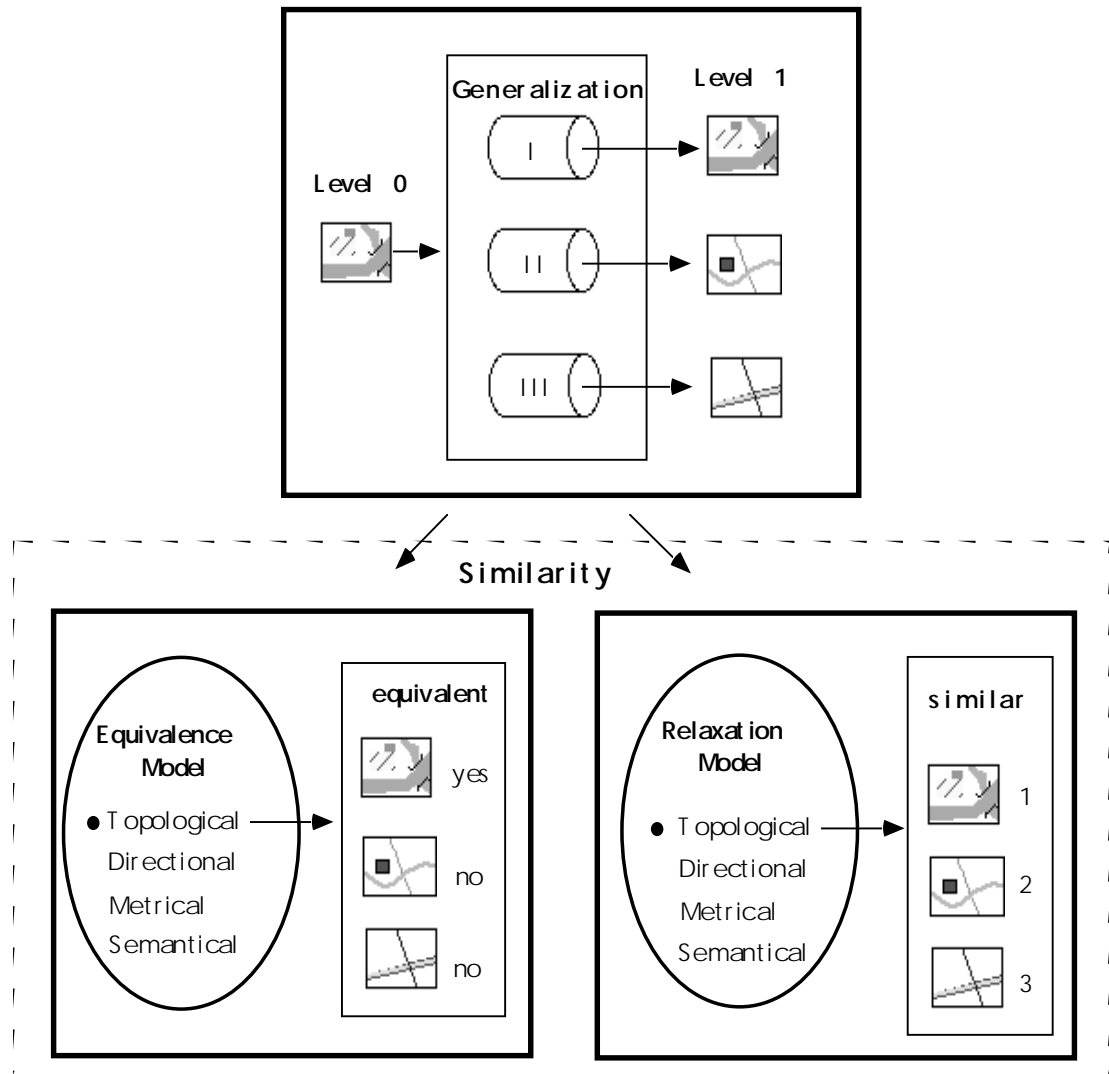


Figure 1.4: Multiple representation framework.

1.4 Motivation

Users have diverse needs that require geographic data at different levels of detail. Cartographic maps at different scales are examples of applications that need multiple representation levels. One critical point in geographic databases with multiple levels of detail is to maintain the topological consistency between the spatial objects. The term consistency is abstract and depends on the constraints applied. In this thesis, equivalent

representations are considered consistent, and less similar representations do not necessarily are inconsistent. Inconsistencies among multiple representations may be fatal as high-level decisions that were based in one model of geographic reality are passed down to detail planners who have contradictory information at hand, or vice versa. In this case, recommendations are made with map information that does not agree with the information available at the decision level.

Current GISs lack methods to maintain consistent multiple representations of geographic objects. There has been research in recent years on various aspects of multiple representations. Some of them are related to data models (Bruegger and Frank 1989; Timpf et al. 1992), cartographic generalization (Buttenfield 1991), and modeling and querying (Rigaux and Scholl 1994), but all of them exclude the analysis of the topological consistency of objects related with their spatial relations. It is important to have consistent object characteristics through different levels of representations, to allow a query at a coarser level to give the same, or at least a very similar result as the evaluation on a detailed level.

1.5 Hypothesis

The initial research efforts concerning multiple representation databases in general differ from the main objective of this thesis, which is to develop a qualitative model to support the implementation of a topological checker to evaluate equivalence and similarity in a multi-resolution spatial database. Consistency across multiple representations refers to the lack of any logical contradictions within a model of reality. Databases with multiple levels may be a result of complex transformations, called generalization operations in cartography. Known generalization operators in the literature include simplification,

smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement, and displacement (Shea and McMaster 1991). Each transformation reduces the complexity of a representation level and generates a representation level that is at most as general as the original representation. Each representation level represents a spatial scene, which corresponds to a set of objects that are related by topological relations, distance relations, and direction relations. Given two spatial scenes, they are considered equivalent if the spatial relations between all the object representations and the structure of individual complex objects are preserved. The scenes are considered similar if the spatial relations and object structure are somewhat equivalent.

The hypothesis of this thesis is: “A qualitative spatial data model for evaluating topological equivalence and similarity in multi-representation databases simplifies the processing of topological queries in a GIS”. The hypothesis is proven by developing a topological qualitative model that supports the multiple representations of spatial objects, and by designing and implementing an algorithm to identify equivalence and similarity between different representations. The major task is to identify what are the components of this qualitative model that supports the development of tools to verify topological equivalence or similarity between spatial scenes, based on a set of possible transitions for the different types of spatial representations.

1.6 Approach

To prove the hypothesis, this thesis develops a new formalism to model spatial objects and spatial relations between several objects, that may be represented at multiple levels of detail. This work focuses on the topological consistency constraints that must hold among the different representations of objects; however, it is not concerned with generalization

operations related to deriving one representation level from another (Beard and Mackaness 1991; McMaster and Shea 1992). Topological consistency is considered at a higher level, independent of the way that spatial objects are encoded. Usually, topology in GIS is concentrated at the conceptual level of nodes, lines, and areas (Corbett 1979; Herring 1987; Egenhofer et al. 1989) and the topological consistency is treated by counting the number of arcs and nodes to guarantee that a map topology is complete (Laurini and Milleret-Raffort 1992). This method is appropriate to evaluate topological structure changes by metric changes on the object. However, it does not capture relations among the objects and sometimes changes in the topology, such as a change in the dimension of a topological element, the aggregation of several parts into a single object, or the elimination of holes. Other works in multiple representation have focused on cartographic line generalization (Buttenfield and McMaster 1991) and algorithms to derive a coarser line from a line with more detail (Douglas and Peucker 1973; Muller 1990; Beard 1991; Wang and Muller 1993).

The term spatial scene in this thesis applies to the spatial object representations for some geographic area. These representations may be points, lines, or regions. Every line has two end points (just one end point if closed line), and each region is composed by a list of connected lines or just one line. The set of lines and points of the spatial scene form a planar graph, and the regions of the scene correspond to faces of the graph.

The proposed approach to identify topological equivalence or similarity between spatial scenes consists of representing the spatial scene as a hierarchical graph model and then applying an algorithm to identify isomorphism configurations between graphs. Current GISs organize the information about objects using the definitions of the standard for digital cartographic data (NCDCDS 1988). This data structure stores the points, lines,

and regions of a spatial scene with additional topological information that make it possible to derive adjacency and connectivity relationships between objects, without needing to use the geometric information. This structure is formally defined as cell complexes (Bruegger and Kuhn 1991), in which an object is represented by a set of cells describing their interior and boundary elements. The cell complex theory has been extended to represent complex objects, such as regions with holes (Puppo and Dettori 1995). This thesis proposes to organize a spatial scene directly as a relation-based model that uses a graph representation. The focus of the relation-based model is on relevant object representations. Only the highest-dimensional object representations are explicitly represented, while their parts are not represented in the graph. The relation-based model stores all connectivity relations, making it possible to answer topological queries more efficiently than the cell complex structure. The relation-based structure is a better cognitive model as it represents the natural meaning people would give to features in a spatial scene. Cell complexes, on the other hand, are a better computational model composed of building blocks that are put together, with the objective of representing the topology of the scene from a computational point of view.

In this thesis, a spatial scene is described by a graph representation composed of a set of graphs describing connected elements and isolated elements. The graph nodes represent the objects and their attributes, while the graph arcs store the spatial relationships between these objects. The topological equivalence between two scenes is evaluated through a scene matching process, which tries to find a one-to-one correspondence between elements of both scenes. A perfect match between the object representations of two scenes identifies two isomorphic configurations. Isomorphism between graphs representing spatial scenes means that both representations can be

topologically equivalent, and the isomorphism occurs if and only if there is a one-to-one mapping of all graph nodes (object representations) such that all adjacent relationships are preserved. The simple fact of finding isomorphic configurations does not guarantee that the scenes have the same topology. Once a one-to-one mapping is found, the boundary sequence of intersections for each object representation is analyzed in order to validate these mappings.

1.7 Results

The result of this thesis is a comprehensive formalism to evaluate topological consistencies across multiple representations. An object-oriented prototype in C++ has been developed as a checker to assess topological equivalence and similarity between spatial scenes composed by objects with multiple representations. The method employed is based on an existing categorization of topological relations (Egenhofer and Herring 1991), and is extended to cope with legal and illegal geometric changes across the multiple representations. This new theory supports consistent topological changes for different configurations of objects like points, lines, and regions, and complexly structured objects such as regions with holes, objects with separations, and heterogeneously composed objects. The result of this thesis work is important in the process of developing multi-representation GISs as it frees databases developers from the tedious task of manually comparing geographic databases that are represented at different scales, and finding discrepancies among the different representations. It also enables database designers to test whether the implementations of new generalization operations perform as desired. Besides, the qualitative information for the spatial scenes can be used as spatial metadata description in digital libraries. Meta information in a library allows

users to retrieve and use data. A subset of a spatial digital library should include spatial concepts such as multiple representations and spatial relations (Beard and Smith 1997). A set of topological similarity measures between object representations and isomorphic configurations is derived in this thesis. These similarity measurements and the qualitative information can be used in digital libraries to allow users to retrieve spatial data.

1.8 Intended Audience

This thesis is intended for anyone involved in working with design of spatial database systems. These include researchers involved with geographic database design, cartographic generalization, as well computer scientists who are concerned with implementation aspects of object-oriented models in GIS.

1.9 Thesis Organization

The remainder of this thesis is organized into six chapters.

Chapter 2 reviews previous work in the area of topological relations and multiple representations in GIS. The description involves topics related with database design of multi-scale databases, links between multiple representations of data, topological consistency for spatial objects with multiple representations, and general constraints that should be incorporated into generalization processes. Characteristics of some generalization algorithms are presented and analyzed in terms of how they preserve the general structure of the data.

Chapter 3 describes the technical background necessary to understand the formalism of working with legal and illegal changes of topological relations. It focuses on the concepts of homogeneity of individual elements and the homogeneity of spatial

relations between objects. The basic definition of topology is presented as well the notion of point-set topology. The topological-relation model with its content and component invariants is described as a basis to model the spatial scene in the form of a graph representation. Finally, some basic concepts of graph theory applied to scene matching are presented, with the main objective of clarifying the concepts of isomorphism and homeomorphisms.

Chapter 4 develops a relation-based model to represent spatial scenes. This chapter concentrates on describing the characteristics of this model, and making a comparison with the cell complexes structure that is widely used in some GIS implementations. Cell complexes represent the geometry of spatial objects through points, lines, and areas and derive spatial relationships from coincidence and inclusion. Qualitative spatial models abstract away the details of the geometry and focus primarily on the spatial relations among objects by modeling them explicitly. While cell complexes are tailored to cartographic representations, the relation-based model captures geometry without requiring a map-like representation. The relation-based model is the basis for the implementation of the topological checker between two different spatial scenes.

Chapter 5 describes the necessary elements to check topological equivalence or topological similarity between spatial scenes. Topological equivalence is analyzed in terms of individual objects, simple or complex, and then in terms of the complete scene. The rules to guarantee equivalence are relaxed by evaluating the conceptual neighborhood of the component invariants of spatial relations. With a new set of constraint rules we are able to identify not just equivalence between graphs, but also to identify levels of similarity between two configurations.

Chapter 6 describes the implementation aspects of the topological checker developed. The class structures and operations for the graph model are introduced. A more detailed description of the attributes and operations can be found on Appendix. This chapter also describes how to integrate this qualitative representation model into the GIS software called SPRING (INPE/DPI 1997), developed by the National Institute of Space Research (INPE 1997). This software organizes the vector data as a cell complex structure.

Chapter 7 concludes this thesis work with further considerations. The results are analyzed, as well as the main contributions of this thesis. Possible future work is also mentioned.

Chapter 2

Multiple Representations

The National Center for Geographic Information and Analysis (Abler 1987) began the discussion of objectives and the process of developing a research agenda in multiple representation databases in the late 1980s (Buttenfield 1989). The research addressed the need to formalize object descriptions at different levels of detail, and to formalize how to link these different levels such that changes applied at one level can carry over to others, allowing multiple representations to be deduced automatically. In terms of cartographic generalization, the research addressed the need that generalization algorithms should incorporate additional constraints in order to preserve the general structure of the objects and to keep them consistent through different levels. As a result of this research, the conceptual focus on automating scale-change operations and map simplification moved toward a formalization of the entire cartographic design process. Categories of research developed from this initiative include such topics as scale-dependent geometry (Buttenfield 1989), digital terrain issues (Weibel and DeLotto 1988), map generalization (Mark 1989), hierarchical data structures (Bruegger and Frank 1989), formalizing databases links (Bruegger and Kuhn 1991), and conceptual frameworks for geographical knowledge (McMaster 1991). Some recent work in this field has been concerned with

modeling and querying multi-resolution databases (Rigaux and Scholl 1994; Puppo and Dettori 1995), multiple paradigms for automating map generalization (Ruas 1995), data and knowledge modeling for generalization (Ruas and Lagrange 1995), an object-oriented model to handle multiple representations (Kidner and Jones 1994), development of multi-scale structure to link representations (Devogele et al. 1996), database design for multi-scale GIS (Jones et al. 1996), and consistency among multiple representations of spatial data (Egenhofer et al. 1994; Tryfona and Egenhofer 1997). The remainder of this paper describes some previous work related to multiple representations.

2.1 Database Design

Database issues in multiple representations are concerned with how to accommodate the different sources of information in a single or multi-version data management strategy. These multiple sources of information need to be maintained consistently, and it is important to organize the multiple topological and metrical information for efficient access and to implement links between these multiple representations (Buttenfield 1993). A transformation from a large-scale map to a small-scale map involve changes that may introduce inconsistencies among representations, which may affect the answers to queries. Due to the inadequacy of automated generalization software, it is infeasible at this time to store a single representation and then to derive other representations that satisfy user queries at specific scales (Beard 1988). As a result, multiple representations for the same data exist, and we expect consistency among them. Multi-scale databases generate multiple representations of data, and some main goals during the design of a multi-scale database are (Jones et al. 1996):

- maximize the database integrity with multiple representations;
- reduce the need of interactive intervention in update operations; and
- automate the retrieval of spatial information relating to phenomena with multiple representation.

2.1.1 GEODYSSEY Database Design

A conceptual design for a multi-scale database has been proposed and implemented by Jones et al. (1996). This system uses object-oriented, deductive, and procedural programming techniques to maintain database integrity to reduce manual intervention of user and to efficiently retrieve spatial information. [Figure 2.1](#) shows the components of the GEODYSSEY system. The intensional database contains rule-based and procedural knowledge required to perform updates and queries, while the extensional database contains stored data items. Semantic, temporal, and spatial data, are represented in the extensional database. The Real-World Object Directory (RWOD) records all real-world objects in the database and provides links to their geometric representations to facilitate the process of updating the geometry. The metadatabase contains information about the quality of the stored geometry. The geometry may be organized into a single or multi-resolution data structure, and it is a component of the spatial database block. The spatial relations are derived from the geometry. The intensional database contains spatial, temporal, and non-spatial rules to support the processes of updating and querying. The database integrity of the geometric representation uses attributes such as minimum bounding rectangle, dimensions, line length, anchor line length, and bandwidth, in order to match multiple geometric representations during the process of updating or querying the database. The topological consistency for spatial relations is implemented on a system

called MAGE (Bundy et al. 1995), in which the generalization is applied on large scale maps that are represented by a triangular data structure called Simplicial Data Structure. The triangulation topology is maintained after applying generalization operators such as simplification and enlargement.

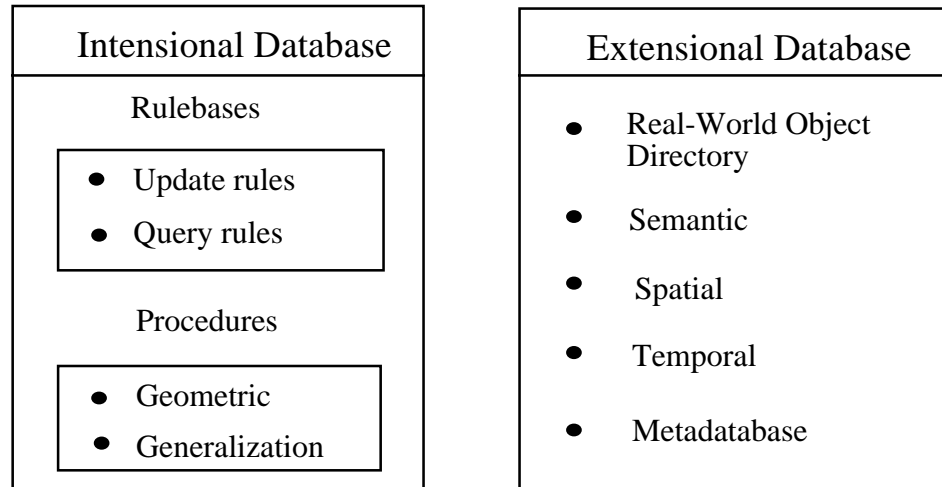


Figure 2.1: GEODYSSEY multi-scale database design.

2.1.2 Links Between Representations

Most current Geographic Information Systems use only one level of abstraction to represent the real world. This single level of abstraction reduces the system utility as topological queries can take a long time to be answered due to the great amount of data stored. Solutions have been proposed in order that GISs support multiple levels of abstraction by linking the different representation levels through hierarchical relations.

Bruegger and Kuhn (1991) introduced the largest homogeneous cells (LHCs) to support the hierarchical structure to link multiple representations. In GIS the concept of cells is used to characterize objects with different dimensions. A 0-dimensional cell represents a point, a 1-dimensional cell represents a line, and a 2-dimensional cell

represents a region. LHC is defined as the largest possible set of cells that is able to distinguish between the interior, the boundary, and the exterior of objects. The proposed approach consists of several single topological representations featuring different levels of abstraction, and the link between these single representations is achieved by connecting the set of LHCs of one topological representation with the set of LHCs of another one. The transition between a less detailed representation to a more detailed representation refines the cell contents. A n -dimensional cell at a less detailed representation may be represented by m -dimensional cells at a more detailed level, with m varying from n to 0. Therefore, at a more detailed representation, a 2-dimensional cell may be represented by a set of 2, 1, and 0-cells; a 1-dimensional cell may be represented by a set of 1 and 0-cells; and a 0-dimensional cell is again a 0-cell.



Figure 2.2: Multiple topological views for Brazil: country, regions, states.

Figure 2.2 shows a structure for multiple topological representations to organize the political subdivision of Brazil. The first level contains the country boundary, the second

level incorporates the country regions, and the third level incorporates the region states. The single 2-cell of the higher level is decomposed into several cells, as the representation is refined.

2.2 Automated Generalization

The need to keep the consistency at different levels of detail in a multiple representation database puts additional constraints on the process of automating cartographic generalization. Cartographic generalizations require the application of both spatial and attribute transformations in order to maintain data clarity and appropriate content for a resultant scale. Digital generalization includes intrinsic objectives (why we generalize) (McMaster and Shea 1988), situation assessment (when we generalize), and spatial and attribute transformations (how we generalize) (Shea and McMaster 1991). The process of how to generalize as defined by Shea and McMaster (1991) corresponds to a set of generalization operators to be applied in maps in order to solve possible spatial conflicts. Operators to reduce the number of objects, spatial operators, attribute operators, and display operators have been proposed to specifically respond to conflict resolution (Beard and Mackaness 1991). Those operators are referred to as *structural operators*, which simplify or abstract the level of detail, and *display operators* which adjust the graphic display to ensure legibility. [Figure 2.3](#) shows some generalization operators (Shea and McMaster 1991) that may affect the general topological structure of the data. Aggregation corresponds to joining a group of different features into a higher-order feature. Collapse means to represent a feature as a lower-order feature, such as representing a region as a line or a line as a point. Amalgamation joins features of the same class into a larger element of this class. Simplification and smoothing may change the general shape of an

individual line, which may cause changes in the relationship of this line with other components of the map.

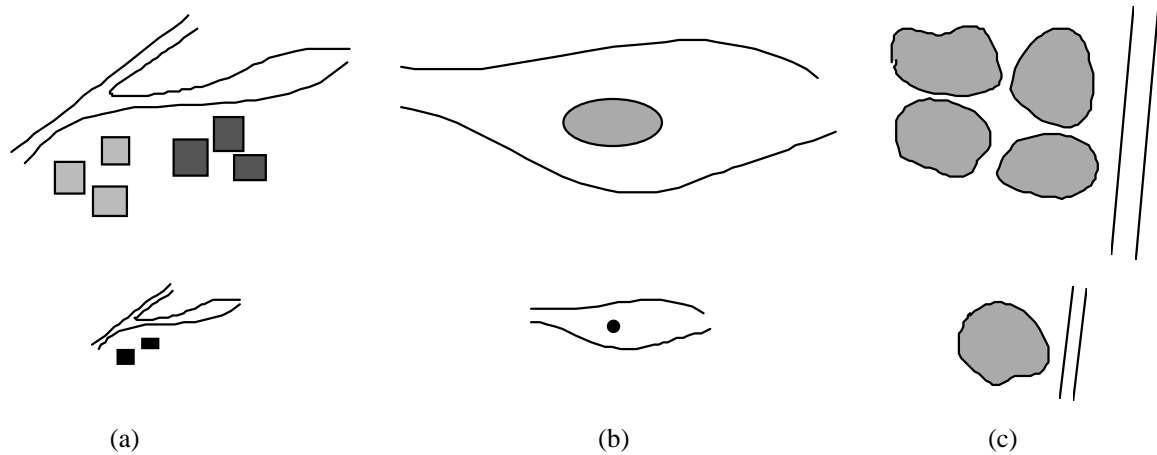


Figure 2.3: Some generalization operations: (a) aggregation, (b) collapse, and (c) amalgamation.

Generalization can be viewed as a set of metric transformations on the geometric representations of spatial objects, intended to improve data legibility and understanding. It is also viewed as an interpretation process that leads to a higher level view of some phenomena (Muller et al. 1995). These two different views have motivated the concepts of cartographic generalization that deals with geometric information, and model generalization or model-oriented generalization that deals with the development of data models to support spatial data at multiple scales and level of details. Model generalization abstracts away the geometric aspects of the spatial data. Some work related to model generalization has been proposed to support updates and queries for different scales (Becker et al. 1991), as well as to develop data structure for “scaleless” geographic databases (Oosterom 1989) and hierarchical structures to divide and merge data with generalization purposes (Jones and Abraham, 1986). More recently Ruas and Lagrange

(1995) presented some work related with data and knowledge modeling for generalization, and Battenfield (1995) presented an object-oriented solution to multi-scale data modeling using the Digital Line Graph (DLG-E) data model developed at U.S Geological Survey (Guptill 1990). Techniques of Artificial Intelligence have also been used to implement rule-based generalization systems, which incorporate geometric knowledge, structural knowledge, and procedural knowledge (Armstrong 1991; Muller 1991).

2.2.1 Spatial Knowledge for Generalization

The development of automated software for cartographic generalization must contain a formal description of the conceptual framework for digital generalization, a set of procedures and generalization operators, and a set of cartographic knowledge rules (McMaster 1991). The integration of expert systems with object-oriented technology has been proposed for object representation and software development (Mark 1991). Generalization incorporates subjective components that do not readily decompose into logical rules. Rules developed for one application (e.g., a network) may not be applicable to another (e.g., a vegetation map). The spatial and attribute relationships between objects should be considered and they can be very diverse, which complicates the process of generating rules. Integration of multiple paradigms such as topology, geometry, hierarchical partitioning, and local triangulation, has been proposed to automate generalizations based on conflict detection and resolving (Ruas 1995).

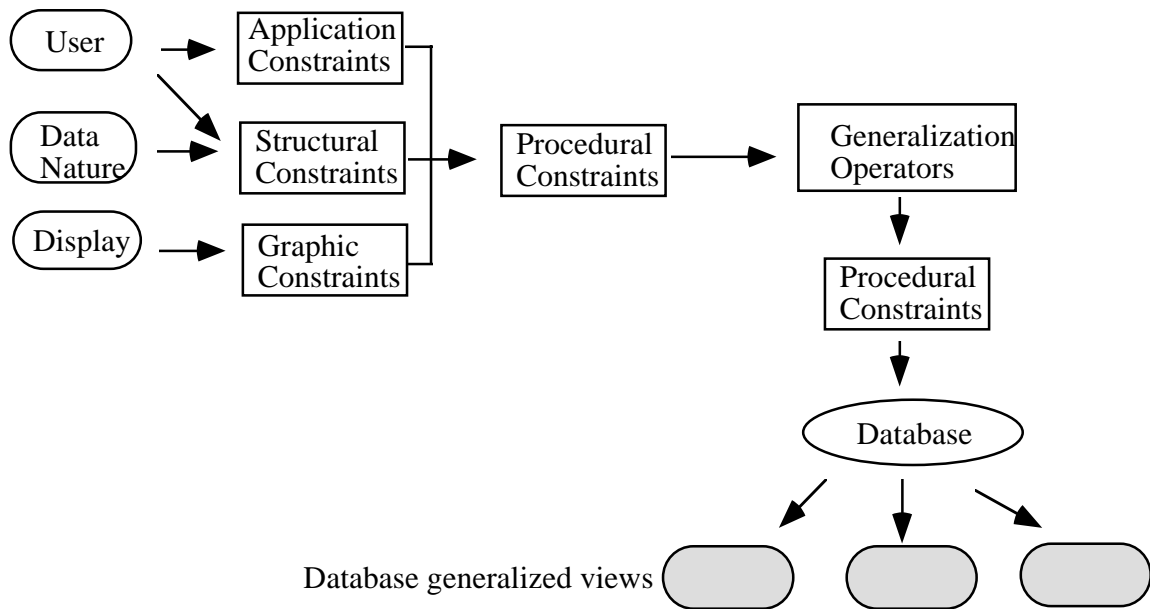


Figure 2.4: Constraint-based framework (Beard 1991).

Since it is difficult to generate rules for generalization, Beard (1991) proposed an interactive approach based on the use of additional constraints as substitutes for rules (Figure 2.4). The *a priori* constraints are derived from generalization controls, while the remaining constraints can be specified interactively by users and varied to reflect different objectives or purposes. These types of constraint include graphic constraints derived from display configurations, structural constraints such as spatial relationships and attributive values, application constraints related with a specific map purpose, and procedural constraints to control the order of an interaction of operations. Conflicts can occur between the structural, graphic, and application constraints, and procedural constraints are used to control the order and interaction of operations and the order in which the constraints are satisfied.

Brassel and Weibel (1988) proposed a model for automated generalization incorporating techniques of expert systems. The rules and procedures for generalization

are stored in a process library, which contains operators, knowledge, and tolerance values. Five processes are presented in this framework, and their connections are represented in [Figure 2.5](#):

- Structure recognition: identify specific cartographic objects or aggregates, as well as spatial relations, and measures of importance. Controlled by original database quality, target map scale, and communication rules.
- Process recognition: identify generalization operators.
- Process modeling: compile rules and procedures to apply from the process library.
- Process execution: execute generalization operators.
- Data display.

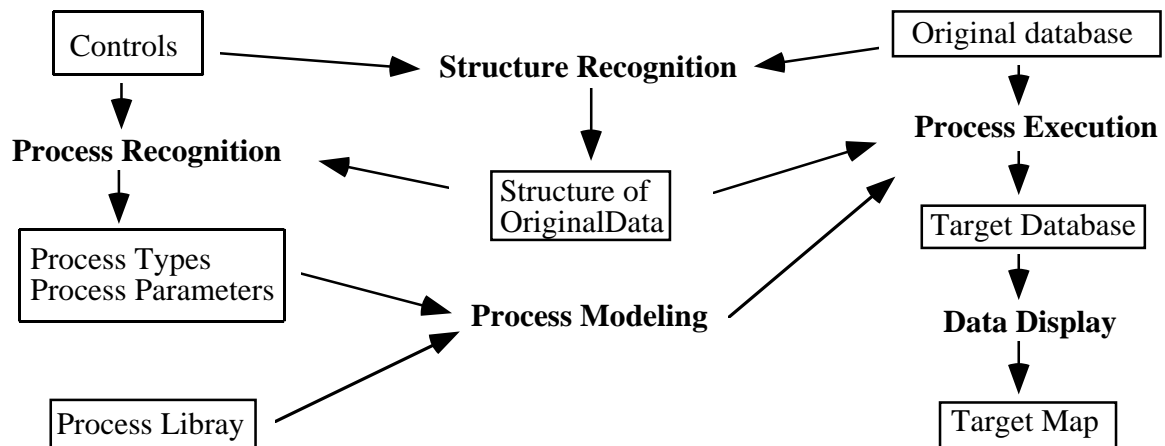


Figure 2.5: Framework for automated generalization (Brassel and Weibel, 1988).

2.2.2 Model Generalization

Model generalization is concerned with map reduction and derivation of spatial data at multiple levels of accuracy and resolution. This different view of generalization has produced the classification of generalization operators that are model-oriented or not (Dettori and Puppo 1996, Ruas and Lagrange 1995). As in cartographic generalization, different applications of model generalization will require different methods. Weibel (1995) specified some general requirements that should be met by all procedures for model generalization:

- Produce predictable and repeatable results;
- minimize deviations from original model;
- maximize data reduction;
- do not violate topological consistency of spatial objects;
- minimize procedure complexity; and
- minimize computations.

Ruas and Lagrange (1995) presented a detailed study about data and knowledge modeling for generalization. Generalization should be seen as a process allowing us to perform a change in the perception level of geographic data, and it must preserve as much as it can the geometric properties, spatial relations, and semantic relations while respecting graphic limitations. Related to geometric properties are needs to identify specific algorithms for generalization based on the characteristic of the objects, to describe the linear elements based on the nature of the objects that they represent, and to apply geometric modeling for the whole linear element or just part of it. Related to spatial relations are issues of connectivity and spatial arrangement relations. Connectivity

relations should be preserved to propagate shifting and deformation between connected objects, and to allow identification of objects that share identical local geometry. In terms of spatial arrangement relations, constraints should be applied in order to preserve proximity and geometric distribution of objects, and tools such as detection of spatial conflicts, elimination and aggregation of objects, should be provided to keep the data topological structure. Related to semantic relations the modification of object geometry may generate conflicts that may be solved through aggregation, elimination, or change in dimension (collapse) of objects. Besides, the notion of simple and complex objects should be established in order to enable efficient modeling of objects. [Table 2.1](#) shows topological generalization operators and the equivalent object properties and relations.

| Operation | Geometry | Connectivity | Proximity / inclusion | Semantic Properties |
|------------------|--|--|---|---|
| selection | T: elimination of too small objects | T: selection of objects which implement a connectivity link | | T: Importance of object depends on their nature |
| aggregation | C: Maintenance of characteristic local shapes. The geometric class of the resulting object has to be the same as before | C: Topologic changes must be restricted to aggregated objects T: Adjacent objects may be aggregated | C: The new objects must be contained in the same face T: Close objects may be aggregated | C: Objects of similar nature may be aggregated T: Components of a complex object can be aggregated |
| collapse | T: Detection of small objects | C: Topology update | C: Update of structure of proximity relations | T: Definition of applicable symbolization |
| displacement | C: Maintenance of characteristic global shapes T: Irregular shapes have to be moved first | C: Angles of sections must be maintained T: Propagation throughout the network | C: Maintenance of distribution structures T: Propagation on close objects | C: Important objects are displaced by others |

Table 2.1: Topological generalization operators and object properties and relations (Ruas and Lagrange, 1995). T= tool, C= constraint.

Buttenfield (1995) proposed an object-oriented multi-scale data model based on the extended Digital Line Graph (DLG-E) data model developed at U.S Geological Survey (Guptill 1990). It extends the DLG-E model with multiple representation schemes for a single record within a single database, providing links between these representations. The multi-scale extension follows the hierarchical and object-oriented form of the DLG-E model. Each object contains spatial attributes and non-spatial attributes that refine the object definition. Relational links between objects are defined based on the dimensionality of the objects, and they allow feature definitions for compound objects. Scale variations may cause changes on geometry and topology, and it is important to note that a model to support transformations between scales of 1:X to 1:Y may not be appropriate for changes between scales 1:Z to 1:W.

2.2.3 Cartographic Generalization

Map generalization is usually associated with line generalization. Line generalization refers to the reduction of the original number of points in a line in order to simplify the data at lower scales. The basic criteria for point reduction includes minimizing displacement and distortion, minimizing new vertices, and minimizing the computational complexity. A standard algorithm for line generalization is the work developed by Douglas and Peucker (1973), which produces some good results, however, due to the lack of topological control, it may produce self-intersecting lines when it eliminates several points from the original line. Several new methods try to incorporate procedures to avoid self-intersections and to keep the general data structure (Muller 1990; Li and Openshaw 1992; de Berg et al. 1995).

Most of the line simplification algorithms analyze only the line points and do not take into consideration that a line may be part of a polygonal subdivision, which is the common approach of commercial GIS. Categorical maps, such as political boundaries or land use, are made up of a group of polygons. A recent work developed by Weibel (1996) attempts to identify further constraints in relation to established cartographic principles in order to form a basis for the development of extended line generalization algorithms that try to preserve the general structure of the data. Four different types of constraints are discussed in terms of an individual line, in terms of a feature class (represented by polygon), and in terms of different feature classes:

- Metric constraints: mainly influenced by aspects of perceptibility such as minimal separation, minimal size, or minimal width.
- Topologic constraints: maintenance of topologic consistency, including avoidance of self-intersections, mutual overlaps, containment of point features.
- Semantic constraints: relates to semantic modeling, preservation of class memberships, or the domain of existence in the spatial context.
- Gestalt constraints: can only be met if the other constraint types are satisfied. Maintenance preservation of original line character or of the distribution and arrangement of map features.

| CONTEXT | CONSTRAINTS | |
|------------------------|--|-----------|
| | Description | Type |
| Within a line | avoid small crenulations | metric |
| | avoid self-coalescence | metric |
| | minimize shape distortion | metric |
| | avoid self-intersections | topologic |
| | preserve line character | gestalt |
| Within a feature class | minimum polygon size | metric |
| | minimum polygon width | metric |
| | preserve ratios between polyg. | metric |
| | avoid line intersections | topologic |
| | preserve categorical class | semantic |
| | maintain visual balance | gestalt |
| Within feature classes | preserve ratios between feature classes | metric |
| | preserve proximity relations | metric |
| | preserve polygon containment | topologic |
| | preserve shared lines | topologic |
| | preserve domain of spatial context | semantic |
| | preserve interplay of elements | gestalt |

Table 2.2: Weibel's (1996) constraints for cartographic generalization.

Table 2.2 summarizes the basic constraints defined by Weibel (1996) and identifies which types of constraints affect an individual line, an individual feature class, or a set of feature classes. Constraints within a line include avoiding very small crenulations, keeping distance between consecutive bends inside a minimum tolerance (avoid coalescence), preserving line length and angularity, preserving line as a polyline, and preserving the original line characteristic. Constraints within a feature class include

keeping the polygon size compatible with minimum area (eliminate or exaggerate), keeping polygon width above minimum distance, preserving area ratio of each category, avoiding intersection between lines, preserving polygon adjacency, connectivity and containment, preserving polygon attributes, and maintaining the overall map pattern. Constraints between feature classes include preserving the ratios between classes, preserving distance relations (parallel lines, point in polygon), preserving polygon hierarchy, preserving shared boundary lines, preserving the domain of existence in the spatial context, and maintaining the interrelationships between the elements.

2.2.3.1 Linear Feature

There are several algorithms developed to simplify lines, and many authors divide them into different categories. Basically these algorithms apply to each individual line separately and do not take into consideration the context of the lines in a map representation nor do they avoid self-intersections. McMaster (1989) divided the algorithms into five categories: (1) independent point algorithms, which do not take into account the mathematical relationship between neighboring pairs of points; (2) local processing routines, which utilize the characteristics of the immediate neighboring points to determine selection/rejection; (3) constrained extended local processing routines, which use distance, angles, or number of points to search beyond neighbor points; (4) unconstrained extended local processing routines, which use the geomorphologic complexity of the line to search beyond neighbor points; and (5) global routines, which consider the entire line or specified line segment.

Hierarchical methods for line generalization have been proposed in order to speed up the visualization of maps at different scales (Cromley 1991). The binary line generalization tree (BLG-tree) stores the result of the Douglas-Peucker (1973) line generalization algorithm in a binary tree. Figure 2.6 shows an original line and its binary tree. The original polyline consists of the points p_1 to p_n , and the most coarse approximation is the line segment $[p_1, p_n]$. The next approximation is defined by the point that has the largest perpendicular distance to segment $[p_1, p_n]$, which generates two new segments $[p_1, p_k]$ and $[p_k, p_n]$. This process is recursively applied with the new segments until all points are inserted in the tree. This structure speeds up the process of retrieving a line for an specific scale, but it is not concerned with other features that are part of the whole map.

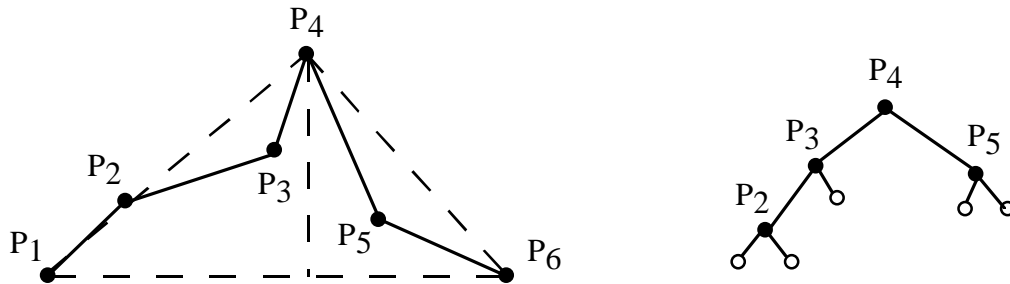


Figure 2.6: Original line and its BLG-tree.

2.2.3.2 Area feature

Usually the generalization algorithms are applied to linear features, but they are not taken into consideration if the linear feature is part of an area feature. An automatic generalization approach for area features has been proposed by Müller and Zeshen (1992), with data display objectives for different scales. This method includes the following steps: data pre-processing, area expansion and contraction, elimination, reselection, aggregation, displacement, topological integrity check, smoothing, and

reduction. The generalization rules defined put emphasis on larger patches over smaller ones, and try to preserve the overall pattern of the relationships between objects by keeping the topological integrity. Data pre-processing determines which patches are to be expanded or contracted by ranking the patch sizes. The elimination process erases all areas below a pre-defined tolerance, and some areas are reselected afterwards as they may be part of a significant cluster. Aggregation between features occurs due to overlaps generated by expansion process. Elimination and merging lead to change in topology. Figure 2.7 shows some acceptable and unacceptable changes in topology considering that all area patches represent the same type of object. Neighboring patches are displaced if they are separated by a distance too small to be visible after expansion. Finally the contour of patches are smoothed using the triad angle thinning algorithm (Zygor 1984).

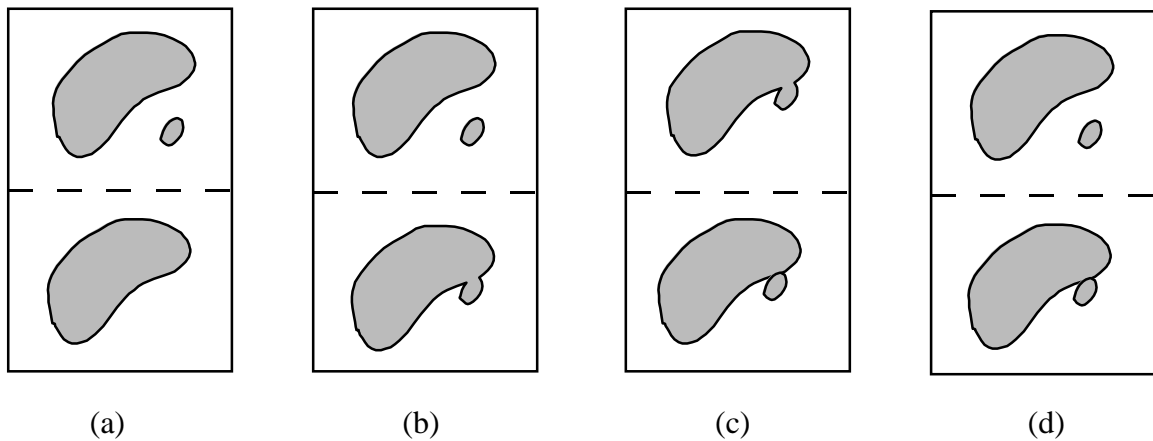


Figure 2.7: Acceptable topological transformations (from top to bottom): (a) object elimination, (b) object aggregation—and unacceptable topological transformations—(c) object addition, and (d) connectivity addition.

2.2.3.3 Graph Approach

In map generalization the appropriate use of generalization operators such as merging, exaggeration, and selection requires information at the geometric and attribute levels. Graph theory has been used in several areas of geographical analysis such as transport planning (James et al. 1970) and spatial reasoning about flow directions in river networks (Paiva et al. 1992). Mackaness and Beard (1993) have described the use of graph theory to support map generalization. The graph representations provide the topological information necessary for effective application of generalization operators. The graph-theoretic approach allow us to detect and to preserve the topological characteristics of map objects such as adjacency, connectivity, and isolation.

A graph corresponds to a set of vertices and a set of edges that connect those vertices. The degree of a vertex is the total number of edges connected at this vertex. For a directed graph, the out-degree of a vertex is the total number of edges leading out from it, while the in-degree is the total number of edges leading toward it. An auxiliary vertex in a graph has degree 2 with edges of the same type (for example, same river separated into two pieces). A weighted graph has attribute values assigned to arcs or even to vertices. These graph properties of weight assigned to graph elements, connectivity between edges, and direction of edges have immediate use in generalization. Weights assigned to a stream network by using Strahler's stream-ordering model (Strahler 1960) can be used to define a rule for lakes. If a lake before the generalization has a degree greater than zero then after generalization the degree should remain greater than zero (assuming that the lake still exists). Connectivity between edges helps in identifying a simplified representation in which the objects are still connected. Directed graphs permit to collapse two edges of the same type connected at an auxiliary vertex.

Based on the graph properties, Mackaness and Beard (1993) defined a set of cartographic generalization rules and described how the graph theory could be used to aid in applying the following generalization operators:

- Simplification: remove auxiliary vertices, maintaining connectivity.
- Aggregation: identify group of the same type.
- Merging-refinement: select essential network components, maintaining connectivity.
- Displacement: identify dense areas, check consistency after.
- Selection: identification of information that is contextual due to the local proximity to salient features.
- Exaggeration: identification of features in isolation or in possession of topological characteristics requiring cartographic emphasis.

2.3 Consistency Among Representations

Multiple representations of spatial data encompass changes in the original geometric representation of objects. The objects are represented from basic geometric primitives such as points, lines, and polygons. Homogeneous objects are represented by one or more primitives of the same type, while more complex structured objects may be represented by a combination of different geometric primitives. These several representations for the same object at different levels should be maintained consistently in order to avoid contradictions in the database. Contradictions would lead to erratic behavior as queries for which a user expects to receive the same result, may produce different results. Consistency must preserve the topologic, distance, direction, and semantic properties of individual objects, as well must preserve the spatial relations between them.

Egenhofer et al. (1994) have proposed a framework to assess topological consistency of multiple representations. The topology of any object and any topological relation between objects must stay the same or continuously decrease in complexity and detail through different levels. This approach uses the model for topological relations (Egenhofer and Herring 1990), with its content invariants (empty/non-empty) for the intersections between the boundary, interior, and exterior of objects, and with its component invariants that describe in more detail the boundary-boundary intersections between the objects (Egenhofer and Franzosa 1995). The necessary component invariants to consider are the sequence of boundary-boundary intersections, the boundary type, the boundary dimension, and whether a boundary is next to a bounded or unbounded exterior union of objects. The consistency is based on the concept of homeomorphism representations. Given two topological spaces X and Y related by a function $f : X \rightarrow Y$, and if the function f and its inverse $f^{-1} : Y \rightarrow X$ are continuous, then this function f is called a *homeomorphism*. Two representations are object-homeomorphic if the general structure of the objects is preserved. In [Figure 2.8a](#) the topology of all corresponding objects at different levels is the same; therefore, the generalization is object-homeomorphic. Two representations are relation homeomorphic if the general structure of the spatial relations between the objects is preserved. In [Figure 2.8b](#) the topological relations between all objects have been retained, therefore, the generalization is relation-homeomorphic. An object-homeomorphic representation does not necessarily preserve the spatial relation between objects, as well as a relation homeomorphic representation does not necessarily preserve the object structure. Different degrees of similarity are described by more or less deviations from the homeomorphism concept. Object similarity and relation similarity are formally expressed as changes in the component invariants that

can be measured, such as boundary-boundary intersection dimensions, number of boundary-boundary components, and number of holes for regions with holes (Egenhofer et al. 1994). For spatial relations between objects, the boundedness of a component invariant may sometimes change its value.

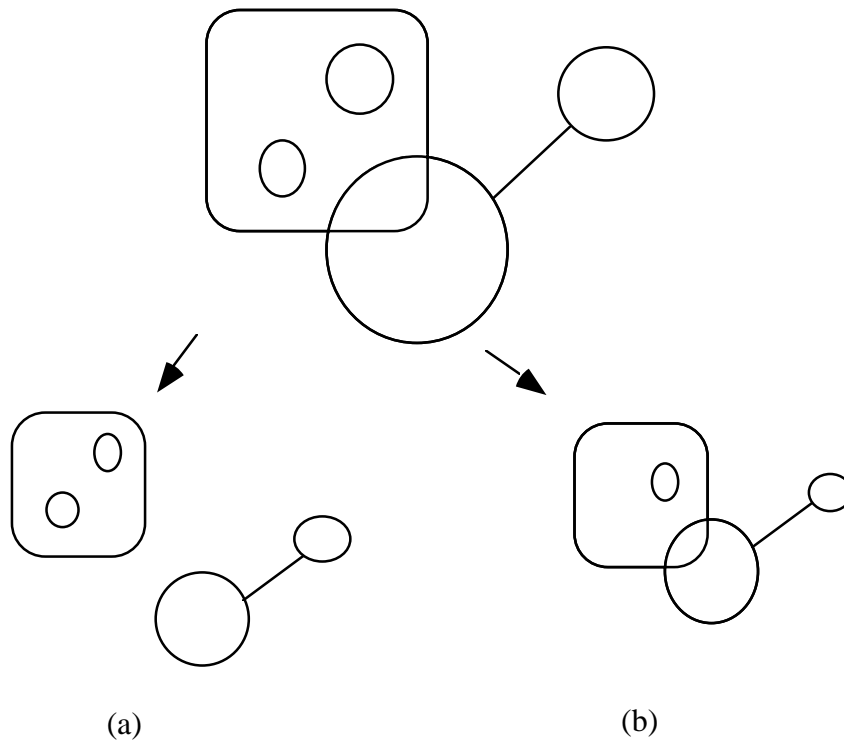


Figure 2.8: Homeomorphism concept: (a) object homeomorphic and (b) relation homeomorphic.

2.4 Summary

Topological consistency in multi-representation geographic databases is one of the major goals of database design. The users expect to get consistent answers for their queries independent of the level of representation of the spatial data. The different levels of representation should be linked, as well as the presence of integrity rules is important

during the process of updating or querying the database. It is also relevant that generalization operators analyze the objects not just as individual entities, but take into consideration the relationships between these objects in order to preserve the general representation of the spatial data. The following chapter describes the mathematical background to assess the topological consistency in multi-representation spatial databases.

Chapter 3

Mathematical Background for Assessing Topological Consistency

This chapter reviews the technical background used in this thesis to assess topological consistency in geographic databases with multiple representations. A database with multiple representations of data means that the same object is represented in several different ways depending on the scale of analysis. Consistency means the lack of any logical contradiction within a model of reality. Two representations for the same spatial region are considered topologically consistent if the topological relationships fulfill certain consistency constraints. The representations can be equivalent if the topological relations are preserved, as well they can be somewhat similar where they satisfy some similarity constraints. The theories that support this thesis are based on a model for binary topological relations (Egenhofer and Herring 1990; Egenhofer and Franzosa 1995), and on graph theory used for scene matching (Ranganath and Chipman 1991). The remainder of this chapter describes the general concepts of topology (Munkres 1975), as well as general concepts about the theory of topological graphs (Gross and Tucker 1987).

3.1 Topological Space

Topology is the study of the properties of geometric figures that are not normally affected by changes in size or shape. A *topology* on a set X is a collection T of subsets of X having the following properties:

- The empty set \emptyset and set X belong to T ;
- The union of the elements of any subcollection of T is in T ;
- The intersection of the elements of any finite subcollection of T is in T .

A set X for which a topology T has been specified is called a *topological space*. A topological space is an ordered pair (X, T) consisting of a set X and a topology T on X . A subset U of X is considered an *open set* of X if U belongs to the topology T . Therefore, a set X with a collection of subsets of X called open sets, represents a topological space. A subset A of a topological space X is said to be a *closed set*, if the set difference $X - A$ is open. Open sets represent the sets in a topology on X , while the closed sets are their complements. The collection of closed sets is closed under arbitrary intersections and under finite unions, and it contains the empty set and X .

3.2 Point-Set Topology

Point-set topology is a theory applied to topological spatial relations between sets in which the relations are defined in terms of the intersections of the boundaries and interiors of two sets (Alexandroff 1961). Using the concept of open sets defined in the previous section, a set-theoretic notion of closeness can be established. A set U is said to be a *neighborhood* of element x , if U is an open set and x belongs to U . The definitions of

interior, closure, and boundary of an element are based on the concepts of open and closed sets.

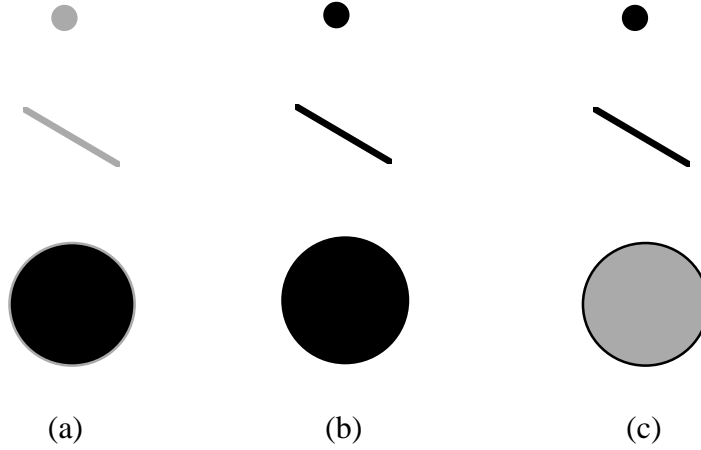


Figure 3.1: Point, line, and region: (a) interior; (b) closure; (c) set-theoretic boundary.

3.2.1 Interior

Given a set element X , the interior of X is defined as the union of all open sets that are contained in X , which corresponds to the largest open set contained in X . The interior of set X is denoted by X° . An individual element x is considered in the interior of X if and only if there is a neighborhood of x contained in X , i.e., x belongs to the interior of X ($x \in X^\circ$) if and only if there is an open set U such that x belongs to U , and U is contained in X ($x \in U \subset X$). [Figure 3.1a](#) shows the interiors for the geometric primitives point, line, and region.

3.2.2 Closure

The closure of a set X is defined as the intersection of all closed sets that contain X , which represents the smallest closed set containing X . The closure of X is denoted by \bar{X} . An element x is in the closure of X if and only if every neighborhood of x intersects X , i.e., x

belongs to the closure of X ($x \in \bar{X}$) if and only if for every open set U containing x , the intersection between U and X is not empty ($U \cap X \neq \emptyset$). [Figure 3.1b](#) shows the closures of a point, a line, and a region.

3.2.3 Boundary

The boundary of a set element X , denoted by ∂X , corresponds to the intersection between the closure of X and the closure of the complement of X ($\partial X = \bar{X} \cap \overline{Y - X}$) and it is a closed set. An element x is in the boundary of X if and only if every neighborhood of x intersects both X and its complement. [Figure 3.1c](#) shows the set-theoretic boundaries of a point, a line, and a region.

3.2.4 Relationships between interior, closure, and boundary

The previous concepts of interior, closure, and boundary are the basis for the definition of topological spatial relations between sets. The relationships between these three elements are:

- the intersection between the interior of a set and its boundary is equal to the empty set ($X^\circ \cap \partial X = \emptyset$); and
- the union of the interior of a set with its boundary results in its closure set ($X^\circ \cup \partial X = \bar{X}$).

3.3 Cell Complexes

Algebraic topology (Alexandroff 1961) is a branch of geometry that deals with the algebraic manipulation of symbols that represent geometric relationships and their

relationships to one another. The algebraic-topology spatial data model is based on primitive types, called *cells*, which are defined for different spatial dimensions:

- a 0-cell represents a 0-dimensional object,
- a 1-cell is the link between two distinct 0-cells, and
- a 2-cell is the area described by a closed sequence of non-intersecting 1-cells.

A face of a n -cell A is any $(0...n)$ -cell that is contained in A . The closure of an n -cell A is the set of all $(0...n)$ -faces that are contained in A . The set-theoretic boundary of a n -cell A is the union of all $(0...n-1)$ -faces that are contained in A . The interior of a cell A is the set difference between A 's closure and A 's boundary. The exterior of a cell A is the set of all cells in the universe that are not elements of the closure. The combination of primitive cells generates more complex ones, called *cell complexes*. By embedding all cells into the same universe it is possible to perform topological operations on a purely symbolic level, without any consideration of metric if the topological structure fulfills the following two completeness axioms (Frank and Kuhn 1986):

- Incidence: The intersection of two cells is either empty or a face of both cells (no two objects must exist at the same location). The 1-cell representing the adjacent boundary of two 2-cells is recorded only once;
- Inclusion: Every n -cell is a face of a $(n+1)$ -cell. A 0-cell is either a start or an end node of a 1-cell, and every 1-cell is in the boundary of a 2-cell.

3.4 Topological Homeomorphism

Given two topological spaces X and Y , a function $f : X \rightarrow Y$ is said to be *continuous* if for each open subset U of Y , the set $f^{-1}(U)$ is an open subset of X . The function f is said

to be *injective* if there is a one-to-one correspondence of elements of X in Y , in which each pair of distinct elements of X have different images under f . The function f is said to be *surjective* if every element of Y is the image of some element of X under the function f . If f is both injective and surjective, then there is a one-to-one correspondence between X and Y , and the function f is called *bijective*.

The inverse of function f is denoted by $f^{-1} : Y \rightarrow X$, and if f is a bijective function, and both f and f^{-1} are continuous, then the function f is called a *homeomorphism*. For a homeomorphic representation, each open set U of X has an equivalent open set in Y under the inverse mapping. The concept of a homeomorphism in topology means that the general topological structure of some data is preserved under some types of transformation. Topological homeomorphism is analogous to the notion of isomorphism in algebra. Isomorphism is a bijective correspondence that preserves the algebraic structure. Figure 3.2 shows two topological homeomorphic configurations at different scales. The generalization process G preserved the topological structure and there is a one-to-one correspondence between the elements.

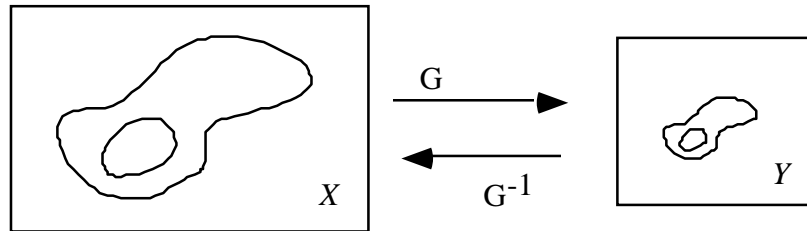


Figure 3.2: Topological homeomorphism.

3.5 Topological Relation Model

The topological relation model (Egenhofer and Herring 1991) formalizes the different topological relations that may exist between spatial objects. Topological relations are

preserved under topological transformations such as translation and rotation. Topological information is a qualitative property and excludes any consideration of quantitative measures. For example, two objects are considered neighbors if they share a common boundary, but the neighborhood relationship is independent of the boundary length. The topological model formalizes the concept of adjacency and containment between objects, which is described by a set of intersections between the interior, boundary, and exterior of these objects. Topological transformations do not necessarily preserve direction and distance relations. Spatial objects may be represented by primitive geometric features such as regions, lines and points. These primitives are usually referred to as cells, which are defined for different spatial dimensions.

3.5.1 Content Invariants

The binary topological relation R between two cells A and B is based on the comparison of the interior, boundary, and exterior of each cell. More specifically, the six object parts are combined to define the matrix of intersections, which characterizes the spatial relation between two cells. The interior of cell A is denoted by A° , the boundary by ∂A , and the exterior by A^- . The matrix of intersections between two cells is defined as the 9-intersection matrix ([Equation 3.1](#)).

$$R(A, B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \quad (3.1)$$

Different combinations for the intersection values describe different topological spatial relations. The topological relations are characterized by topological invariants of

the nine intersection values, i.e., some properties that are preserved under topological transformations. The content invariant which expresses emptiness or non-emptiness for the intersection value, is the most general topological invariant (Egenhofer and Franzosa 1991) and it characterizes each of the nine intersections by a value empty (\emptyset) or non-empty ($\neg\emptyset$).

The nine empty/non-empty intersections describe a set of relations that provides a complete coverage. Five hundred-twelve topological relations are possible between two regions; however, only some of them can be realized in a 2-dimensional space. Figure 3.3 shows the nine intersection matrices for the eight topological relations that can be realized between simple regions.

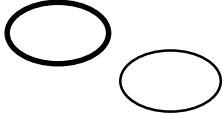
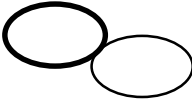

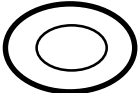

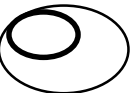
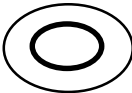

| | | | |
|---|--|---|---|
| <p>disjoint</p>  $\begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$ | <p>meet</p>  $\begin{pmatrix} \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$ | <p>overlap</p>  $\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$ | <p>contains</p>  $\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$ |
| <p>equal</p>  $\begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \emptyset & \neg\emptyset & \emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$ | <p>coveredBy</p>  $\begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$ | <p>inside</p>  $\begin{pmatrix} \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \emptyset & \emptyset \\ \neg\emptyset & \neg\emptyset & \neg\emptyset \end{pmatrix}$ | <p>covers</p>  $\begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$ |

Figure 3.3: Eight topological relations between two regions in \mathfrak{R}^2 (Egenhofer and Herring, 1991).

The content invariant is too general to capture topological differences in more complex configurations. In this cases other types of topological invariants need to be considered.

3.5.2 Component Invariants

The 9-intersection model with its content invariant of empty and non-empty intersections is a comprehensive model—the resulting topological relations provide complete coverage and are mutually exclusive—but it is sometimes too generic to support the different views people might want to make.

If the topological relation between A_X and B_X in X is equivalent to the topological relation between A_Y and B_Y in Y , then the associated 9-intersections are the same, but conversely, the same 9-intersection set does not mean that the topological relations are the same. In order to assess whether two topological relations between two objects are the same or not, additional topological invariants are considered for simple regions, based on the boundary-boundary intersection components (Egenhofer and Franzosa 1995). A boundary-boundary component is a separation in the boundary. [Figure 3.4](#) shows a configuration with a *meet* relation and three boundary-boundary components. The *boundary-boundary components sequence* describes the order in which the components of the boundary-boundary intersection occur. The equivalence of two boundary-boundary sequences, $S1$ and $S2$, is obtained when the sequence $S1$ matches with at least one of all sequences of $S2$ obtained by its cyclic permutation. Therefore, the component invariants that will be necessary to distinguish topological details between connected elements are (assuming that the space is partitioned and no overlap occurs):

- The *boundary-boundary sequence* captures the order of the boundary-boundary components of the boundary-boundary intersection. It requires an agreed-upon orientation of the plane—clockwise or anti-clockwise. For example, the clockwise boundary-boundary sequence for object B in [Figure 3.5a](#) is 1, 2, 3.
- The *boundary dimension* represents the dimension of each boundary-boundary component. A point intersection has dimension 0, and a line intersection has dimension 1. For [Figure 3.5a](#) the boundary-boundary components labeled 1 and 3 are 1-dimensional, and component 2 is 0-dimensional.
- The *complement relationship* describes how a boundary-boundary component is related to the complement of the union of two regions (*common exterior*). The complement relationship is considered *unbounded* if it touches the *common exterior*, and is considered *bounded* if it does not touch this *common exterior*. [Figure 3.5b](#) stresses the complement relationship of objects A and B : the boundary components 1 and 3 are *unbounded*, while component 2 is *bounded*.

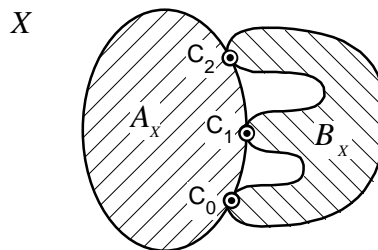


Figure 3.4: A *meet* relation with three boundary-boundary components.



Figure 3.5: Component invariants of topological relations: (a) the boundary-boundary sequence of the dimension and intersection type: $\{(1, 1, \text{meet}), (2, 0, \text{meet}), (3, 1, \text{meet})\}$ and (b) the boundary-boundary sequence of the complement relationship: $\{(1, \text{unbounded}), (2, \text{bounded}), (3, \text{unbounded})\}$.

3.6 Similarity Analysis

A spatial scene corresponds to a set of geographic objects that are related through such spatial relations as topological, directional, and metrical relations. The application of generalization operators in one scene generates a second scene, which in general has a different structure than the original one. Rarely do two scenes match exactly and it is important to evaluate how similar the two configurations are in order to guarantee consistency. Similarity is the assessment of deviation from equivalence (Tversky 1977). A derived scene may have topological relations that are slightly different from the original scene, as well it may have directions and distances that are not exactly the same, and it may have a considerably different shape. Bruns and Egenhofer (1996) proposed a computational method to formally assess the similarity of spatial scenes based on the ordering of spatial relations. The method is based on the concept of gradual changes of spatial relations and it applies to topological relations (Egenhofer and Al-Taha 1992), cardinal directions (Freska 1992), and approximate distances (Hong 1994). Given two

scenes with an equal number of objects, there is a minimum set of gradual changes to transform one scene into another, and the spatial similarity is assessed by counting the number of different spatial relations and by counting the gradual changes in spatial relations.

Topological relations are considered first-class information in spatial scenes, which has to prevail in the case of a conflict between two different representations—topology matters, metric refines (Egenhofer and Mark 1995). The concept of gradual changes has been used to model the conceptual neighborhoods of topological relations (Egenhofer and Al-Taha 1992). This neighborhood concept facilitates an ordering of topological relations and supports the determination of similar relations. [Figure 3.6](#) shows the conceptual neighbors for the eight topological relations for simple objects represented as regions (Egenhofer and Franzosa 1991). A gradual transformation changes a relation into any of its conceptual neighbors, and statements like “covered by is similar to inside” or “meet is more similar to overlap than to contains” can be made.

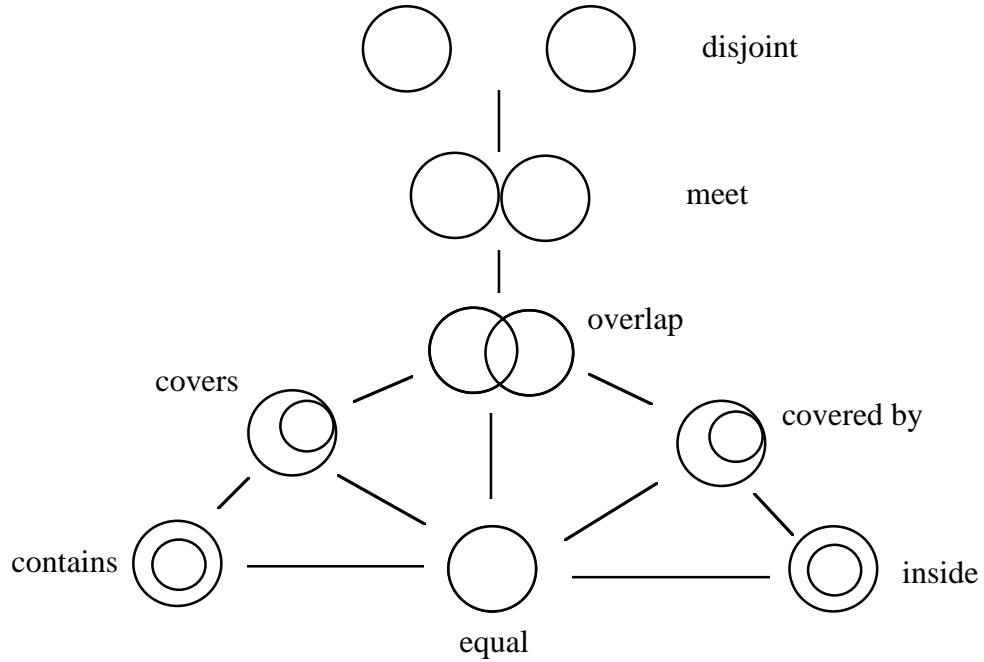


Figure 3.6: Conceptual neighborhoods of topological relations between simple regions (Egenhofer and Al-Taha 1992).

3.7 Graph Model

In topological graph theory, a graph represents a network of nodes and straight or non-straight arcs connecting these nodes. Therefore, a graph G consists of a set of vertices or nodes N and a set of arcs A . Each arc has one or two end points that are equal to some of the nodes in N . A simple graph has no loops or cycles. A graph is said to be *planar* if it can be drawn in a plane so that none of its arcs intersect except at its nodes. Two nodes N_1 and N_2 are said to be connected or adjacent if there is at least one arc in which the end points are equal to N_1 and N_2 . If there is no arc with end points equal to N_1 and N_2 , then N_1 and N_2 are said to be disconnected. Usually an adjacency matrix is used to characterize the connectivity of a graph. This adjacency matrix is square with the number of rows and number of columns equal to the number of nodes in the graph. Each cell of this matrix has the value 1 if the nodes are connected, and is 0 if the nodes are disconnected. Figure

3.7 shows a graph and its equivalent adjacency matrix. Depending on the type of application the nodes and arcs of a graph may carry some specific properties. For example in applications related with network analysis, such as allocation problems or shortest path analysis, the nodes may carry some demand information, while the arcs may carry some impedance values related with the cost to traverse those arcs.

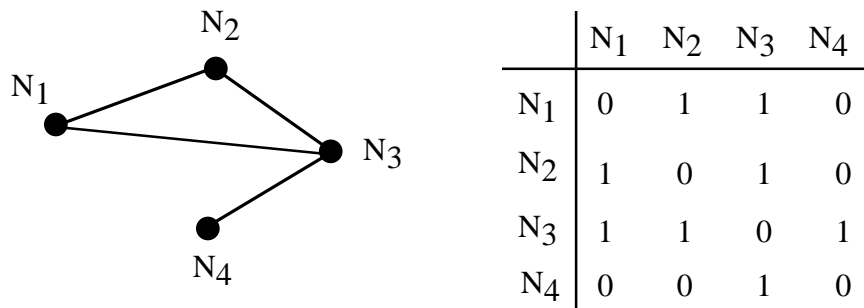


Figure 3.7: Graph and its adjacency matrix.

3.7.1 Isomorphism and Homeomorphism

Two graphs are said to be *isomorphic* if and only if there is a one-to-one mapping between the nodes of the two graphs, such that all adjacent relationships are preserved. Given two graph configurations it is possible to find many isomorphic configurations, but for each configuration the one-to-one mapping is present. Two graphs are said to be homeomorphic if both of them can be combined from the same graph by a sequence of operations that sequentially introduce a new vertex to divide the arc. [Figure 3.8](#) shows a set of graph representations illustrating the concepts of isomorphism and homeomorphism. This concept of isomorphism between graphs is equivalent to the concept of topological homeomorphisms between topological spaces.

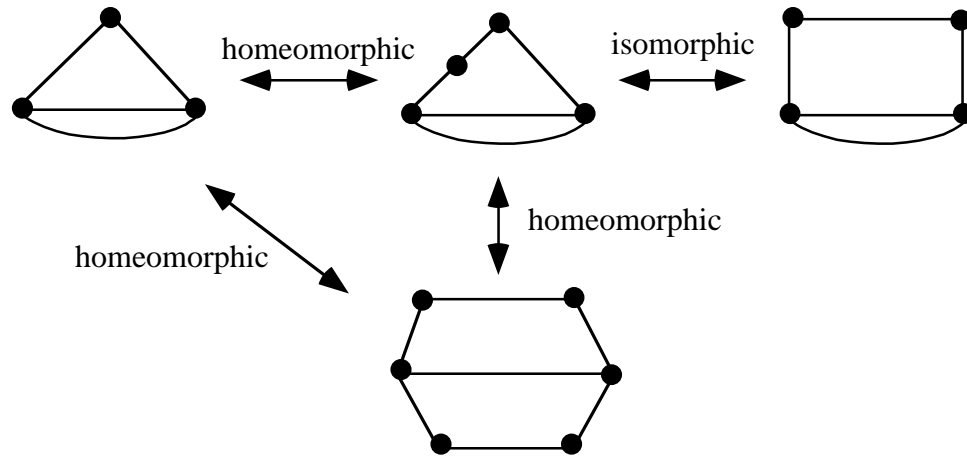


Figure 3.8: Homeomorphic and isomorphic graphs (Chacra *et al.* 1979).

3.7.2 Subgraph

The identification of sub parts of graphs is important in the process of finding similarity between representations. Usually transformations of original data may generate derived data that are not exactly the same as the original data, but they are still consistent. A graph G' is called a subgraph of a graph G , if and only if the nodes and arcs of G' are a subset of the nodes and arcs of G . Figure 3.9 shows a graph and some of its subgraphs.

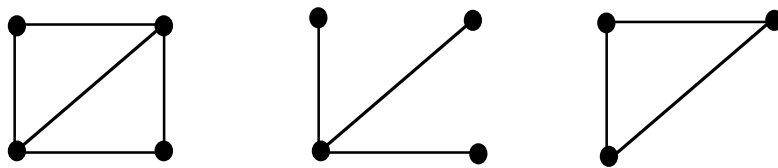


Figure 3.9: A graph and two of its subgraphs.

3.8 Association graph for scene matching

Scene matching is the process of finding a correspondence between elements of two different representations. Some categories of scene matching are template matching methods, feature matching, and graph theoretic matching (Ranganath and Chipman

1991). The graph approach is useful for scale changes and, therefore, fits well in geographic analysis of equivalency and similarity between spatial scenes. It may use the properties of objects and relationships between them to model the graph. A general procedure for graph matching consists of finding a vertex mapping matrix, which indicates all possible equivalent nodes, and then apply a backtracking algorithm to check all possibilities of matching. This approach is used to find isomorphic configurations, and is inappropriate to handle temporal changes.

Another approach to scene matching is called *association graphs*. Given two scenes S_1 and S_2 , the nodes of the association graph correspond to the pair of elements E_{s_1} and E_{s_2} that satisfy some property constraints. The arcs between nodes of the association graph exist if the relationship between the elements of scene S_1 , and the relationship between elements of scene S_2 satisfy the relation constraints. To assess topological equivalency or similarity of spatial scenes, the association graph can be built based on the spatial relations between objects. If the objective is to find equivalent representations, then the nodes and arcs of the association graph correspond to equal elements in terms of topology. Otherwise, there is a problem of inexact matching, in which the relation constraints are relaxed and the association graph represents elements that are equal or at least similar to the relaxation level defined. For example, if there are two spatial scenes composed by regions, we can use the binary topological spatial relations between the regions (Egenhofer and Herring 1991) and the conceptual neighborhood for spatial relations between regions (Bruns and Egenhofer 1996) to build the association graph. [Figure 3.10](#) shows two spatial scenes and the association graph built based on equivalent representations. Each node of the association graph has a pair of elements or object representations that contain the same number of adjacent objects with equivalent

relations, and each arc connects nodes in which the spatial relation for the objects in each scene is the same.

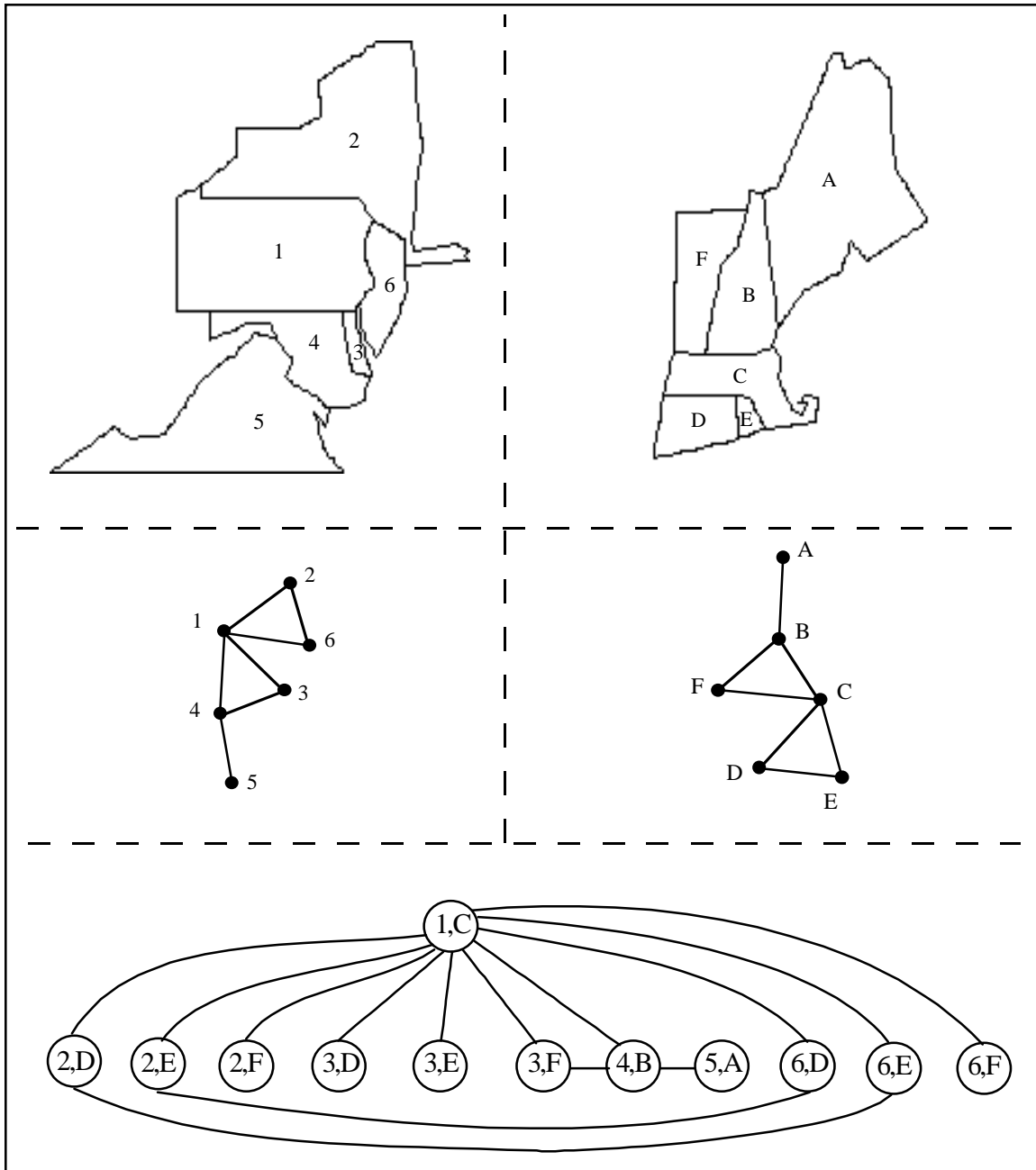


Figure 3.10: Two spatial scenes, their graphs, and the equivalent association graph.

3.9 Summary

Consistency between two data sets depends on the constraint rules that are used to compare both data sets. If two datasets are equivalent or equal, then they are considered consistent, as well if there are differences in those two data sets that are supported by the constraint rules. The topological consistency between two spatial scenes is guaranteed if the objects preserve their general structure, and the spatial relations are preserved or some valid changes have occurred. A graph representation of a spatial scene captures the topological concepts of adjacency and isolation between spatial objects. The association graph is used in this thesis as the basis to start the process of finding isomorphism between the graph representations. By using spatial relations as constraints, the association graph contains just the information that satisfies these constraints and, therefore, it simplifies the number of possibilities when evaluating the isomorphism problem. The pair of nodes represent possible matches and the arcs represent the possible paths to search. The next chapter describes the relation-based model to represent a spatial scene, including all necessary topological invariants that are used to evaluate equivalence between two representations.

Chapter 4

A Qualitative Spatial Model for Multiple Representations

Traditionally, data models for geographic information systems have been map-based, attempting to represent geometry as it would appear on a cartographic map. These cartographic spatial data models are the foundation for digital spatial representations that require a graphical presentation. Such models have been extensively investigated in the GIS literature and have found their way into today's GIS Standards (SDTS 1992; SAIF 1995; OGC 1998). The most sophisticated cartographic spatial data models are based on cell complexes, which originated in algebraic topology (Maunder 1996).

Complementary to cell complexes are representations of spatial configurations and geometry that do not require a graphical presentation. Such models are necessary when no visual information is available, for instance if spatial information was collected from verbal descriptions. They focus on the representation of the spatial relations among objects, rather than on the geometry of individual objects, and are at a higher level of abstraction than the geometric elements of points, lines, and areas. Such relation-based models also enable fast spatial inferences, allow for representations of incomplete

information, and link closely to generating verbal instructions and verbal descriptions of spatial configurations.

Although most current geographic information systems are based on the cartographic model, the linkage to and integration with relation-based models is a pressing need. People usually make abstraction of a spatial representation, thinking about multiple views for real objects and relationships between them. They are more concerned about the relationships between objects than the too detailed information provided by the cartographic model. Besides, relation-based models can be used to represent information from multi-media information systems and multi-modal interactions through voice and sketch (Egenhofer 1996). This chapter introduces the relation-based model, and investigates the mapping from cell complexes onto a relation-based model, and vice versa.

4.1 Relation-Based Model

Qualitative spatial models abstract away the details of geometry and focus primarily on the spatial relations among objects by modeling them explicitly. This section introduces the relation-based model to handle geographic data, which uses the graph representation as basis to represent the spatial scene.

The graph representation provides the topological information necessary for an effective application of generalization operators in spatial configurations. The graph theoretic approach has the ability to detect and to preserve such topological characteristics of map objects as adjacency, connectivity, and isolation. A spatial scene may be described by a graph or by a set of graphs (one for each connected set of elements) in cases where we have disjoint sets of connected elements, or even connected

sets contained or covered by others. The nodes of each graph represent the object representations, and will be referred to as features, and the graph arcs represent the spatial relationships between these object representations. Each node stores the object attributes and each arc represents relationships between objects such as angle, adjacency, or distance. For the purpose of this thesis, the arc attributes represent the topological spatial relations between objects, considering the content and component invariants of spatial relations.

Each graph for a spatial scene is described through a pair $G=(N, A)$, and initially the node and arc properties are formally defined as follows:

- Associate to each node a label by means of a *labeling function* defined as $l_N : N \rightarrow ID \times GR$ that, given a node $N_i \in N$, returns a pair (ID_i, GR_i) , where ID_i denotes an identifier of the entity and GR_i its graphical or geometric representation.
- Similarly we associate a label to each arc by means of the labeling function $l_A : N \times N \rightarrow SR$, that, given an arc $(N_i, N_j) \in A$, returns the spatial relation $SR_{i,j}$ between N_i and N_j .

Figure 4.1 shows a simple spatial scene with its equivalent graph representation. Each feature is mapped into a graph node, and the spatial relations between these nodes are mapped by the arcs that connect them. If two features have more than one boundary in common, then the complete graph must map each common boundary as one arc connecting the elements.

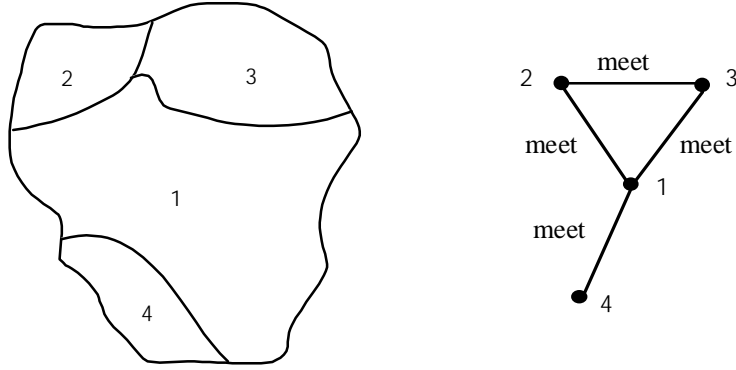


Figure 4.1: A spatial scene and its equivalent graph.

In general a spatial representation of some specific data, such as administrative boundaries, is characterized by a topological structure, in which the topological spatial relations between features are well defined. A spatial scene may contain a connected set of features that share some common boundaries, sets of features disjoint from other elements, as well as features contained or covered by some other element. The next section explains how the topology of a spatial scene may be described by a graph or by a set of graphs. Each graph corresponds to a set of connected features. A hierarchical structure of graphs is specified, such that the descendant features from a higher order area feature are constrained by the topological properties *contains* and *coveredby*, i.e., the higher order feature always must contain or cover its descendants.

4.1.1 Hierarchical Graph Representations

A spatial scene can be described by sets of graphs. These graphs may be composed of disjoint sets of connected features or by sets of features containing or covering others. The former situation is modeled using disjoint graphs, while the latter situation involves spatial relations between the two sets of connected features (namely, one is contained or covered by the other). This section describes how to model such situation. First, each

connected set of features is mapped onto a graph representation. The obtained graphs are then joined through containment links, thereby generating a layered structure. The idea is to represent the containing or covering set and the contained or covered set at different levels in a hierarchy of graphs. Given a graph G that belongs to a level in the hierarchy, the next level contains graphs that correspond to connected sets or even isolated features that are contained or covered by an area feature in G .

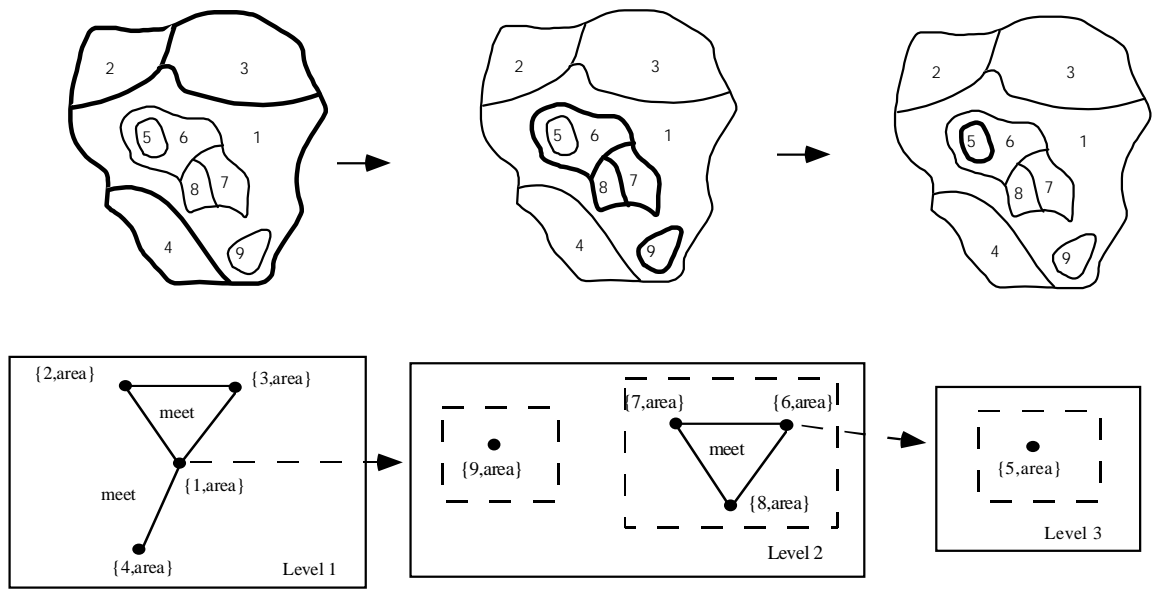


Figure 4.2: Hierarchical levels of a spatial scene and their graph representations.

Figure 4.2 shows a spatial scene in which different levels of a hierarchy can be identified, and the corresponding graph representation. Features 1, 2, 3, and 4 represent one connected set, and this set is in the higher level of the hierarchy. In this example, just feature 1 at the higher level contains some other features, so the second level of the hierarchy is defined by the connected set of features 6, 7, 8 and the isolated feature 9. The third and last level of this scene corresponds just to the feature 5, which is contained by feature 6.

The hierarchical representation can be formally defined in the following way. A hierarchy of graphs HG is a sequence $\{G_{1,1}, \dots, G_{1,n_1}\}, \dots, \{G_{k,1}, \dots, G_{k,n_k}\}$ of k sets of graphs. Each set k represents the number of levels in the hierarchy, such that for each $G_{i,j}$ ($1 < i \leq k$) there exists a graph $G_{i-1,p}$ ($1 \leq p \leq n_{i-1}$) that contains a node N . The region that corresponds to N contains or covers the portion of scene corresponding to $G_{i,j}$. By using this hierarchical structure we are able to reduce the complexity of representations such as region with holes, into simple features that are analyzed at their respective levels. For each level in the hierarchy, the relationships may be modeled by using the content and component invariants applied to the internal holes of an object (Paiva and Egenhofer 1995).

4.1.2 Graph Modeling

The graph modeling uses the concepts of content and component invariants of topological spatial relations presented in chapter 3. [Figure 4.3a](#) shows a set of connected areas and the corresponding graph representation ([Figure 4.3b](#)). Each adjacent boundary corresponds to one arc in the graph representation. The final graph structure can be simplified by abstracting the boundary intersections between two features into just one arc in the graph ([Figure 4.3c](#)). In this case the arc labeling property is modeled by using the boundary sequence of the intersections. By reducing the number of arcs, the complexity of the graph structure is automatically reduced.

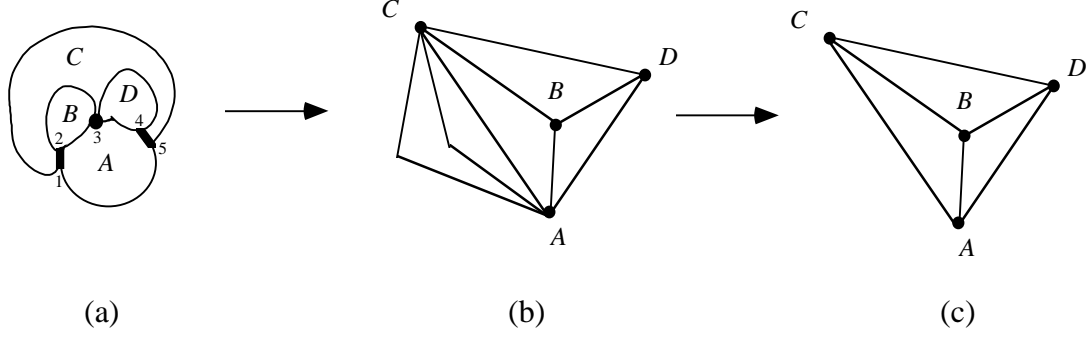


Figure 4.3: Spatial scene and its graph simplifications.

The spatial relation of the arc labeling function $l_A : N \times N \rightarrow SR$ can now be defined as an ordered (for instance, clockwise) sequence of boundary components (Equation 4.1).

$$SR_{i,j} = \left\{ (boundtype_{i,j}, compl_{i,j})_1, \dots, (boundtype_{i,j}, compl_{i,j})_n \right\}$$

where:

i, j = represent the respective nodes (4.1)

n = number of boundary intersections between N_i, N_j

$boundtype$ = boundary type (0 - meet or 1 - meet)

$compl$ = complement relationship (bounded or unbounded)

In the following, the information encoded in the arcs is maintained by defining for each node the sequence in which the intersections occur. This corresponds to the so-called adjacency-lists data structure used for encoding graphs (Aho *et al.* 1974). The formal definition for this node sequence is expressed on Equation 4.2.

$$NS_i = \left\{ \left(N_k, \left(SR_{N_i, N_k} \right)_n \right)_1, \dots, \left(N_k, \left(SR_{N_i, N_k} \right)_n \right)_m \right\}$$

where:

m = varies from 1 to number of boundary intersections (4.2)

k = some node element connected to node N_i

n = one of the boundary components between nodes N_i and N_k

The node sequence set contains the information stored in the arc labeling function, so it is not necessary to store both sets. The spatial scene is completely described in terms of topology by using the node labeling function and the node sequence set. This model is called the relation-based model. It generates a hierarchical graph and is referred to as *HG* model. [Table 4.1](#) shows the equivalent node properties and node sequence sets for the spatial configuration represented in [Figure 4.3a](#). The node definition contains the object identifier and geometric representation, and the node sequence includes the arc properties.

| Node | Def. | Sequence of the boundary components |
|-------|------------|---|
| N_A | {A, area } | $\{(N_C, (I-meet, unbounded)), (N_B, (I-meet, unbounded)),$ $(N_C, (O-meet, bounded)), (N_D, (I-meet, unbounded)), (N_C,$ $(I-meet, unbounded))\}$ |
| N_C | {C, area } | $\{(N_A, (I-meet, unbounded)), (N_D, (I-meet, unbounded)),$ $(N_A, (O-meet, bounded)), (N_B, (I-meet, unbounded)), (N_A,$ $(I-meet, unbounded))\}$ |
| N_B | {B, area } | $\{(N_A, (I-meet, unbounded)), (N_C, (I-meet, unbounded)),$ $(N_D, (O-meet, unbounded))\}$ |
| N_D | {D, area } | $\{(N_A, (I-meet, unbounded)), (N_B, (O-meet, unbounded)),$ $(N_C, (I-meet, unbounded))\}$ |

Table 4.1: Relation-based model HG for [Figure 4.3a](#).

A 2-dimensional feature in a higher order graph may *contain* all features in a descendent graph, or in some cases it may *cover* some of these features. If several cover relations occur, a boundary sequence with these *cover* relations should be added to the HG model. [Figure 4.4](#) shows a scene similar to [Figure 4.3a](#), but with an additional region G that covers region C . [Table 4.2](#) shows the additional information that must be added to [Table 4.1](#) in order to define the HG model.

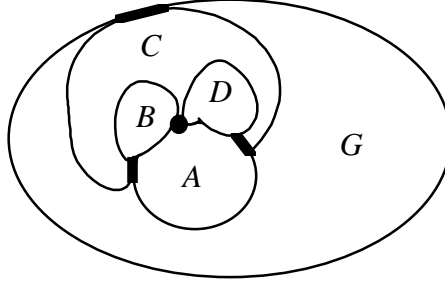


Figure 4.4: A spatial scene with a *cover* relation.

| Node | Def. | Sequence of the boundary components |
|-------|----------------------|---|
| N_G | $\{G, \text{area}\}$ | $\{N_C, (I\text{-cover}, \text{unbounded})\}$ |

Table 4.2: Complementary row to [Table 4.1](#) that describes scene of [Figure 4.4](#).

4.1.3 Adding Symbolic Geometric Information

The main characteristics of the graphs in the hierarchical relation-based model HG are:

- Each graph node corresponds to a cell with order equivalent to the geometrical representation of the graph node. Therefore, if a node represents an area then it corresponds to a 2-cell; if it represents a line then it corresponds to a 1-cell; and if it represents a point then it corresponds to a 0-cell.
- The boundary sequence for one node does not contain any geometric information. This sequence defines the clockwise intersections using only the adjacent features. Therefore, if a 1-cell of a 2-cell is not part of another 2-cell, then any information related to such 1-cell is lost in this model. If a transformation between the HG model to the cell complexes is needed, additional information is necessary in order to restore all elements of the cell complex structure.

In order to get the cell complex representation from the relation-based model HG , spatial coordinate symbols referent to the intersections are added for each adjacent boundary component. For a 0-dimensional boundary, the point coordinate for the 0-cell is added, while for a 1-dimensional boundary both start and end points of the 1-cell are stored. By using those spatial coordinates, it is possible to rebuild the 2-cell using the clockwise sequence of boundary intersections ([Equation 4.3](#)).

$$NS_i = \left\{ \left(N_k, \left(SR_{N_i, N_k} \right)_n, \left(SC_{N_i, N_k} \right)_n \right)_1, \dots, \left(N_k, \left(SR_{N_i, N_k} \right)_n, \left(SC_{N_i, N_k} \right)_n \right)_m \right\}$$

where:

$$SC_{N_i, N_k} \text{ defines the 0-cell coordinates for the adjacent points,} \quad (4.3)$$

and SC will have one coordinate for a 0-dimensional adjacency, and 2 coordinates for a 1-dimensional adjacency.

Each graph augmented with the symbolic geometric information is called an enhanced graph. The obtained hierarchical relation-based model is referred to as the HG^+ model. [Table 4.3](#) shows the HG^+ for the scene in [Figure 4.3a](#).

| Node | Def. | Sequence of the boundary components |
|-------|------------|---|
| N_A | {A, area } | $\{[N_C, (I-meet, unbounded), (1,2)], [N_B, (I-meet, unbounded), (2,3)], [N_C, (O-meet, bounded), (3)], [N_D, (I-meet, unbounded), (3,4)], [N_C, (I-meet, unbounded), (4,5)]\}$ |
| N_C | {C, area } | $\{[N_A, (I-meet, unbounded), (5,4)], [N_D, (I-meet, unbounded), (4,3)], [N_A, (O-meet, bounded), (3)], [N_B, (I-meet, unbounded), (3,2)], [N_A, (I-meet, unbounded), (2,1)]\}$ |
| N_B | {B, area } | $\{[N_A, (I-meet, unbounded), (3,2)], [N_C, (I-meet, unbounded), (2,3)], [N_D, (O-meet, unbounded), (3)]\}$ |
| N_D | {D, area } | $\{[N_A, (I-meet, unbounded), (4,3)], [N_B, (O-meet, unbounded), (3)], [N_C, (I-meet, unbounded), (3,4)]\}$ |

Table 4.3: Relation-based model HG^+ for [Figure 4.3a](#).

4.2 Equivalence Between Cell Complexes and the Relation-Based Model

Given a cell complex C , there exists a cell complex C' that contains the minimal information (in terms of number of cells) to describe the topology of the scene represented by C . In the simplified cell complex representation, the 2-cells are maintained and their boundary is simplified by dropping points and merging lines only if this does not affect the boundary of other cells in the complex ([Figure 4.5](#)). Therefore, a function called *Strip*: $C \rightarrow C'$ maps the set of cells of a cell complex C onto the set of cells of another simplified cell complex C' . The result of this process is a cell complex whose 2-cells are homeomorphic to the cells of C , but whose global number of cells is reduced. Two cell complexes C_1 and C_2 are equivalent if $Strip(C_1) = Strip(C_2)$. The representative element of an equivalence class for a given cell complex C is $Strip(C)$ (called the minimal

element). *Strip* corresponds to the projection function that, given a cell complex, returns the representative of its equivalence class.

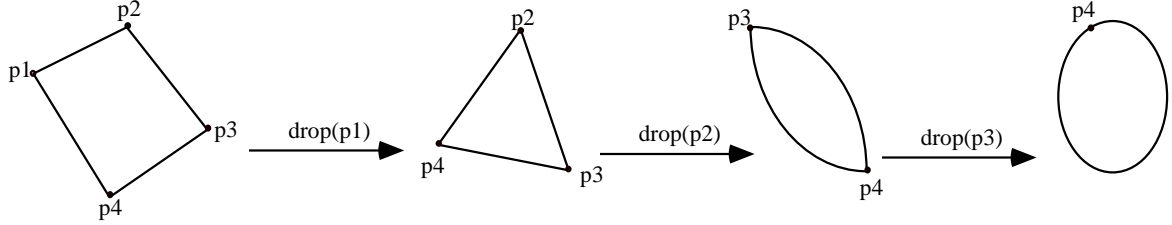


Figure 4.5: Dropping points from a region boundary.

The graph-based representation can be seen as a minimal representation of a spatial scene described by a given cell complex. It is the canonical representative of an equivalence class obtained through an equivalence relation defined on the set of cell complexes. This relation is based on the concept of a homeomorphism between cells. In this way we drop unnecessary information and keep only the minimal cell information in order to describe the topology of the spatial scene. This corresponds to converting a cell complex into a graph structure. If *GraphToCell* is the function that converts a graph structure into a cell complex, and *CellToGraph* is the function that converts the cell complex into a graph representation, then Equation 4.4 should hold.

$$GraphToCell(CellToGraph(C)) = Strip(C) \quad (4.4)$$

4.3 Mapping Cell Complexes into the Relation-Based Model

This section describes an algorithm to transform the structure of cell complexes into the relation-based model. The relation-based model HG^+ is a simplified representation of the cell complexes. Lower-order cells that are part of the boundary of a higher order cell are

not represented in the graph. The graph nodes correspond to higher-order cells. Therefore, the 0-cells of a 1-cell are not in the graph. Likewise the 1-cells of a 2-cell are missing. The 1-cells are part of the graph only if they do not belong to a 2-cell or 1-cell boundary. The 0-cells are part of the graph if they represent isolated points.

4.3.1 Algorithm to Convert a Cell Complex into a Relation-Based Model

Assuming a spatial scene with regions, we may have isolated elements and connected elements. Each isolated element corresponds to a graph with an isolated node, and each set of connected elements corresponds to one graph. Given a cell complex CC with regions, and the regions being hierarchically ordered from higher to lower levels, the basic algorithm to convert this cell complex into the relation-based model is ([Algorithm 4.1](#)):

```

HGraph CellToHGraph (CellComplex  $CC$ )
Begin
  For each region  $R$  of  $CC$ 
    Transform  $R$  into a graph node  $N$  and add to  $hgraph$ ;
    Add  $N$  to the corresponding graph level and set hierarchical links;
    If  $R$  has adjacent element(s)
      Generate boundary sequence for  $N$  from  $R$ ;
  return  $hgraph$ ;
End

```

Algorithm 4.1: Convert cell complex structure into relation-based model
(graph model).

4.3.2 Converting Individual Features

An isolated region, 2-cell composed by 0 (P) and 1 (L) cells, generates a node equal to an area feature in the graph representation $R_i(P, L) \rightarrow N_i$, where N_i is a node such that $l_N(N_i) = (ID, Area)$. Each region of a connected set of regions is a node in the graph representation, and each common line (i.e., line shared by two regions) and each common point (i.e., point shared by two or more regions) between regions defines a boundary component in the boundary sequence for a node. In order to check if two 2-cells share a common point, the information stored in the cell complex is used. First, consider boundary lines of both regions and then retrieve their endpoints (combination of *region-line* and *line-point* relations). Similarly, to get all regions incident at a given point we must combine *point-line* and *line-region* relations. For each region i composed of points (0-cells P) and lines (1-cells L), the equivalent graph node and boundary sequence are generated (Equation 4.5).

$$\begin{aligned} R_i(P, L) &\rightarrow N_i \\ &\rightarrow BoundarySequence \{ \text{clockwise sequence of adj. features of } N_i \} \end{aligned} \quad (4.5)$$

Again $l_N(N_i) = (ID, Area)$.

A 2-cell, composed of n 1-cells has n 0-cells. The boundary sequence components for one region may be generated using Algorithm 4.2.

```

For  $k = 1$  to  $n$  1-cells of a 2-cell
Begin
  For each clockwise 2-cell feature  $S$  that shares the common point  $0\text{-cell}[k]$  with  $R$ 
    Create the boundary description:  $Boundary(0\text{-meet} \times CompRelationship \times XY(0\text{-cell}[k]))$ ;
    Create the boundary component:  $BoundaryComponent(S \times Boundary)$ ;
    Add  $BoundaryComponent$  to boundary sequence of  $R$ ;

  If  $1\text{-cell}[k]$  is part of a 2-cell feature  $R'$  that is different from  $R$ 
    Create the boundary for  $1\text{-cell}[k]$ :  $Boundary(1\text{-meet} \times CompRelationship \times XY(1\text{-cell}[k]))$ ;
    Create boundary component:  $BoundaryComponent(R' \times Boundary)$ ;
    Add  $BoundaryComponent$  to boundary sequence of  $R$ ;
End

```

Algorithm 4.2: Converting the 2-cell boundaries into the graph model.

4.3.3 Example

Figure 4.6 shows a spatial scene with its cells. Applying Algorithm 4.2 to convert the cell complex to the relation-based structure we use the three lists of 0-, 1-, and 2-cells (Equations 4.6, 4.7, and 4.8).

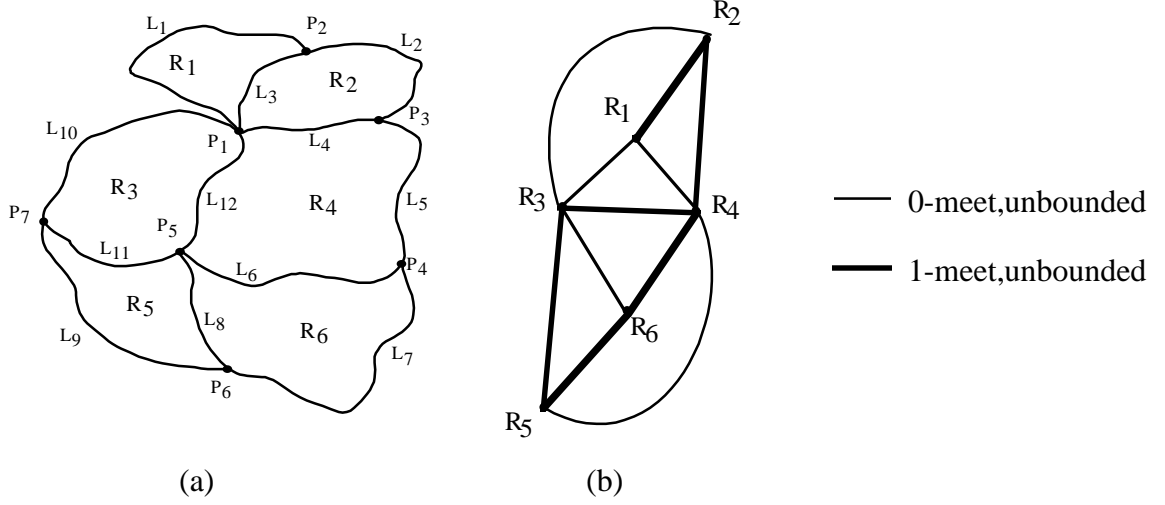


Figure 4.6: Spatial scene: (a) cell features, and (b) graph representation.

$$\mathbf{0-cell} = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7\} \quad (4.6)$$

$$\mathbf{1-cell} = \left\{ \begin{array}{l} [L_1, (P_1, P_2)], [L_2, (P_2, P_3)], [L_3, (P_1, P_2)], [L_4, (P_1, P_3)], \\ [L_5, (P_3, P_4)], [L_6, (P_4, P_5)], [L_7, (P_4, P_6)], [L_8, (P_5, P_6)], \\ [L_9, (P_6, P_7)], [L_{10}, (P_7, P_1)], [L_{11}, (P_5, P_7)], [L_{12}, (P_1, P_5)] \end{array} \right\} \quad (4.7)$$

$$\mathbf{2-cell} = \left\{ \begin{array}{l} [R_1, (L_1, L_3)], [R_2, (L_2, L_4, L_3)], [R_3, (L_{12}, L_{11}, L_{10})], \\ [R_4, (L_5, L_6, L_{12})], [R_5, (L_8, L_9)], [R_6, (L_7, L_8)] \end{array} \right\} \quad (4.8)$$

Next, transforms each region into a graph node of *HG* model ([Table 4.4](#)).

| Cell | Graph Node |
|-----------------------------------|------------------------|
| $[R_1, (L_1, L_3)]$ | $\{R_1, \text{area}\}$ |
| $[R_2, (L_2, L_4, L_3)]$ | $\{R_2, \text{area}\}$ |
| $[R_3, (L_{12}, L_{11}, L_{10})]$ | $\{R_3, \text{area}\}$ |
| $[R_4, (L_5, L_6, L_{12})]$ | $\{R_4, \text{area}\}$ |
| $[R_5, (L_8, L_9)]$ | $\{R_5, \text{area}\}$ |
| $[R_6, (L_7, L_8)]$ | $\{R_6, \text{area}\}$ |

Table 4.4: Equivalent graph nodes for 2-cells of Figure 4.7.

Finally generate the boundary sequence for each region (Table 4.5). The boundary sequence for region R_1 is generated as follows:

Region R_1 is formed by two lines, therefore we have two iterations:

Iteration 1:

Current 0-cell is P_1 :

The clockwise regions that share P_1 with R_1 are R_4 and R_3 then:

Create boundary components: $\{ R_1, (R_4, (0\text{-meet, unbounded}), (P_1)) \}$

$\{ R_1, (R_3, (0\text{-meet, unbounded}), (P_1)) \}$

Current 1-cell is L_1 , and it is just part of region R_1 , then

do not create this boundary component and go to next iteration.

Iteration 2:

Current 0-cell is P_2 :

There are no regions that share P_2 , then do not create any boundary component

Current 1-cell is L_3 , which is part of region R_2 then

Create boundary component: $\{ R_1, (R_2, (1\text{-meet}, \text{unbounded}), (P_2, P_1)) \}$

Iteration 3: end.

Table 4.5 shows the boundary sequence for each region by repeating Algorithm 4.2.

In order to avoid repetitions of boundary components, we can store all boundary components in one list, and access the boundary sequence of each node through pointers to the elements that describe the sequence.

| Region | Boundary Sequence |
|--------|--|
| R_1 | $\{ [R_4, (0\text{-meet}, \text{unbounded}), (P_1)], [R_3, (0\text{-meet}, \text{unbounded}), (P_1)], [R_2, (1\text{-meet}, \text{unbounded}), (P_2, P_1)] \}$ |
| R_2 | $\{ [R_4, (1\text{-meet}, \text{unbounded}), (P_3, P_1)], [R_3, (0\text{-meet}, \text{unbounded}), (P_3)], [R_1, (1\text{-meet}, \text{unbounded}), (P_3, P_2)] \}$ |
| R_3 | $\{ [R_1, (0\text{-meet}, \text{unbounded}), (P_1)], [R_2, (0\text{-meet}, \text{unbounded}), (P_1)], [R_4, (1\text{-meet}, \text{unbounded}), (P_1, P_5)], [R_6, (0\text{-meet}, \text{unbounded}), (P_5)], [R_5, (1\text{-meet}, \text{unbounded}), (P_5, P_7)] \}$ |
| R_4 | $\{ [R_2, (1\text{-meet}, \text{unbounded}), (P_1, P_4)], [R_6, (1\text{-meet}, \text{unbounded}), (P_4, P_5)], [R_5, (0\text{-meet}, \text{unbounded}), (P_5)], [R_3, (1\text{-meet}, \text{unbounded}), (P_5, P_1)], [R_1, (0\text{-meet}, \text{unbounded}), (P_1)] \}$ |
| R_5 | $\{ [R_3, (1\text{-meet}, \text{unbounded}), (P_7, P_5)], [R_4, (0\text{-meet}, \text{unbounded}), (P_5)], [R_6, (1\text{-meet}, \text{unbounded}), (P_5, P_6)] \}$ |
| R_6 | $\{ [R_4, (1\text{-meet}, \text{unbounded}), (P_5, P_4)], [R_5, (1\text{-meet}, \text{unbounded}), (P_6, P_5)], [R_3, (0\text{-meet}, \text{unbounded}), (P_5)] \}$ |

Table 4.5: HG^+ model components for Figure 4.6.

4.4 Mapping the Relation-Based Model onto Cell Complexes

This section defines a function $GraphToCell: HG^+ \rightarrow CC$, mapping an enhanced graph onto the corresponding cell complex (i.e., mapping a set of features and a set of incident/adjacent relations between them). The function maps each node onto a region, line, or point, depending on its corresponding geometric representation. Also, each arc is mapped onto a line or a point depending on the kind of *meet* relation it represents.

4.4.1 Algorithm to Convert the Graph Representation onto the Cell Complex

Each node of HG^+ representing a region generates several cells. The algorithm to convert the whole HG^+ into a cell complex CC ([Algorithm 4.3](#)) basically picks each node and transforms it into a list of cells, and then if this node contains internal elements, then the algorithm recursively converts these internal elements.

```
CellComplex HGraphToCell (HGraph hg)
Begin
  For each node N of the higher level of hg
    Call procedure NodeToCell (N) and add result to cellcomplex;
  return cellcomplex;
End

ListOfCells NodeToCell (Node node)
Begin
  Transform node into a list of cells and add to listofcells;
  For each internal feature I of node
    Call procedure NodeToCell (I) and add result to listofcells;
  return listofcells;
End
```

Algorithm 4.3: Converting the hierarchical graph onto the cell complex.

ListOfCells AreaFeatureToCell (**AreaFeature** R)

Begin

Create *new-2-cell* for R and make *current-0-cell* and *first-0-cell* equal to null;

For each boundary component BC of R

If $BC(\text{dimension})$ equal to 0

If *current-0-cell* equal null

 Create *new-0-cell* equal $BC(0\text{-cell})$ and add to *ListOfCells*;

 Make *first-0-cell* and *current-0-cell* equal *new-0-cell*;

Else

If *current-0-cell* not equal $BC(0\text{-cell})$

 Create *new-1-cell*(*current-0-cell*, $BC(0\text{-cell})$);

 Add *new-1-cell* to *ListOfCells* and to *new-2-cell*;

 Make *current-0-cell* equal $BC(0\text{-cell})$;

If $BC(\text{dimension})$ equal to 1

If *current-0-cell* equal null

 Create *new-1-cell*($BC(\text{start-0-cell})$, $BC(\text{end-0-cell})$);

 Add *new-1-cell* to *ListOfCells* and to *new-2-cell*;

 Make *first-0-cell* equal $BC(\text{start-0-cell})$;

 Make *current-0-cell* equal $BC(\text{end-0-cell})$;

Else

If $BC(\text{start-0-cell})$ not equal *current-0-cell*

 Create *new-1-cell* (*current-0-cell*, $BC(\text{start-0-cell})$);

 Add *new-1-cell* to *ListOfCells* and to *new-2-cell*;

 Create *new-1-cell* (*current-0-cell*, $BC(\text{end-0-cell})$);

 Add *new-1-cell* to *ListOfCells* and to *new-2-cell*;

 Make *current-0-cell* equal $BC(\text{end-0-cell})$;

If *current-0-cell* not equal *first-0-cell*

 Create *new-1-cell* (*current-0-cell*,*first-0-cell*);

 Add *new-1-cell* to *ListOfCells* and to *new-2-cell*;

 Add *new-2-cell* to *ListOfCells* and **return** *ListOfCells* ;

End

Algorithm 4.4: Converting a graph node (region) into a list of cell complexes.

4.4.2 Converting Individual Nodes Representing Regions

Algorithm 4.4 shows in more detail how to convert each node representing an area feature into a list of cells. It follows the clockwise sequence of boundary intersections, and creates the necessary 0-cells and 1-cells that close the region. When there is no connection between two boundary intersections (look at the symbolic coordinates of the boundaries) , a 1-cell is created to link them. These 1-cells correspond to lines of a region that are not shared by other region. Remember that these 1-cells are lost during the conversion of the 2-cell into the graph model.

4.4.3 Example

Applying Algorithm 4.4 for region R_1 of Figure 4.4 we have:

The boundary sequence for R_1 is:

$$\{ [R_4, (0\text{-meet}, \text{unbounded}), P_1], [R_3, (0\text{-meet}, \text{unbounded}), P_1], \\ [R_2, (1\text{-meet}, \text{unbounded}), (P_2, P_1)] \}$$

For the first boundary intersection $[R_4, (0\text{-meet}, \text{unbounded}), P_1]$:

Create 0-cell P_1 and make it the current 0-cell, and the first 0-cell of the 2-cell.

For the next boundary intersection $[R_3, (0\text{-meet}, \text{unbounded}), P_1]$:

P_1 already exists and is the current 0-cell. Therefore nothing to do. Go to next step.

For the next boundary intersection $[R_2, (1\text{-meet}, \text{unbounded}), (P_2, P_1)]$:

P_2 is not equal to current 0-cell P_1 , then create 1-cell(P_1, P_2), and

make current 0-cell equal to P_1 .

Finally, current 0-cell is equal to first 0-cell, then nothing to do and end processing.

4.5 Summary

This chapter introduced a qualitative model for topological representation based on hierarchical graphs, and compared it with the cell complex structure. The cell complex can be seen as a more cartographic model composed of building blocks put together with only a descriptive aim (representation of the topology from a computational point of view). For the graph scene the focus is on “important” object representations. This assumes that some sort of semantics is associated with features in the scene. People make abstractions about the real world. They don’t capture all the details when looking at a spatial scene. There is a need for computational algorithms that behave similarly. The relation-based model captures such properties by focusing on the most important features of a spatial configuration. It disregards such detail as metric and topological issues necessary for drawing a picture. From this point of view the graph structure seems to be a more cognitive model with respect to the cell complex representation. It is of great importance for analyzing scenes at a coarse level. Although the chapter explained the model using polygonal data, the same reasoning can be applied to homogenous and heterogeneous networks. The next chapter shows how this relation-based model can be used in order to check topological equivalence between spatial scenes.

Chapter 5

Assessing Topological Equivalence

In order to compare multiple topological representations and assess whether they contradict each other or not, sophisticated tools are needed. The mechanisms for these assessments differ depending on certain properties of the spatial objects. This chapter uses the relation-based model (Chapter 4) as a basis to develop a computational tool to assess topological equivalence between two spatial scenes. Topological equivalence is analyzed in terms of individual representations for spatial objects, as well as considering spatial scenes composed of a collection of these individual object representations. Although equivalence occurs rarely when some generalization procedure is applied, the equivalence concept can be relaxed in order to support similarity analysis between spatial scenes (Bruns and Egenhofer 1996). The topological equivalence between two scenes is achieved if both scenes have the same hierarchical structure, if the graph representations are isomorphic, and if there is a match between each element of one scene and a unique element on the other scene, respecting the orientation of boundary intersections between the adjacent elements.

5.1 Spatial Objects

Spatial objects may be represented by using the three geometric primitives region, line, and point. These primitives can be used alone or in combination to represent one object. Simple objects have a single geometric representation, while complex objects, such as regions with holes, are represented by a collection of individual regions. Simple objects are topologically equivalent if they are homeomorphic. Complex objects are topologically equivalent if the concepts of object homeomorphism and relation homeomorphism apply (Egenhofer et al. 1994).

5.1.1 Simple Objects

Simple objects contain just one representation, which could be 2-dimensional (region), 1-dimensional (line), or 0-dimensional (point). If after a transformation process the dimension structure of the object representation is preserved, then the two instances of the object are considered equivalent in terms of topology, and they are object homeomorphic.

5.1.2 Relation Between Objects

Spatial relations between objects are topologically equivalent if the content invariants, and the component invariants of spatial relations are preserved (Egenhofer and Franzosa 1995). These invariants are the dimension of the components, their types, their relationships with respect to the exterior, and the boundary sequence of intersections.

5.1.3 Complex Objects - Region with Holes

A region with holes is a homogeneously 2-dimensional object with a connected interior, and disconnected boundaries and exteriors. There exists one *outer boundary* that contains all other boundaries, called the *inner boundaries*. Each inner boundary delimits one

hole—in a degenerate case, a region with one hole, filling the region’s interior, becomes a 1-sphere, i.e., a closed line separating two parts of the exterior from each other (Egenhofer *et al.* 1994).

Holes have to fulfill certain topological constraints: (1) They must be mutually exclusive such that any two holes that are part of the same region cannot overlap, nor may one hole contain another hole. Holes may, however, touch along their boundaries. (2) Holes must be in the region, either fully surrounded by the region’s interior or inside but located along the boundary.

In order to capture these properties, a region with holes is represented as a set of individual, simple regions for which the eight binary topological relations apply that are realized with the 4-intersection. The structure of a region A with holes consists of (1) its generalized region A^* , defined as the union of the region and its holes, and (2) each individual hole H_i^A (Equation 5.1).

$$A = (A^* \setminus (\bigcup_{i=1}^n H_i^A)) \cup (\bigcup_{i=1}^n \partial H_i^A) \quad (5.1)$$

5.1.3.1 Hole Characteristics

Holes have the following characteristics:

- Each hole is a separate entity.
- Since each hole is contained in the generalized region (i.e., $H_i^A \subseteq A^*$), the topological relation that must hold between a hole and the generalized region is *inside* or *coveredBy* or *equal* (i.e., $\forall i: H_i^A \text{ inside } A^*$ or $H_i^A \text{ coveredBy } A^*$ or $H_i^A \text{ equal } A^*$).

- Since any pair of holes cannot have common interiors (i.e., $H_i^\circ \cap H_j^\circ = \emptyset$), the types of topological relations that may occur between holes are *disjoint* and *meet* (i.e., $\forall i, j | i \neq j: H_i^A \text{ disjoint } H_j^A \text{ or } H_i^A \text{ meet } H_j^A$).

The dimension of the relation *meet* between two adjacent holes is based on the boundary-boundary component:

- If the dimension of all boundary-boundary components between two holes is 0, then the topological relation is called *0-dimensional meet* (or simply *0-meet*).
- If the dimension of all boundary-boundary components between two holes is 1, then the topological relation is called *1-dimensional meet* (*1-meet*).
- If the dimension of some boundary-boundary components between two holes is 0, and of some others 1, then the topological relation is called *mixed-dimensional meet* (*mixed-meet*). [Figure 5.1](#) shows a region with holes in which the dimension of the intersections between holes H_2^A and H_3^A is 0 and 1, therefore, they have a *mixed-meet* relation. If there occurs a *mixed-dimensional meet* or an *n-dimensional meet* with more than one component, then an *isolated hole* gets created, i.e., a hole that is *disjoint* from the interior part of the object. In [Figure 5.1](#), the holes H_4^A and H_5^A are examples of *isolated holes* as they have no connection with the interior of object A.

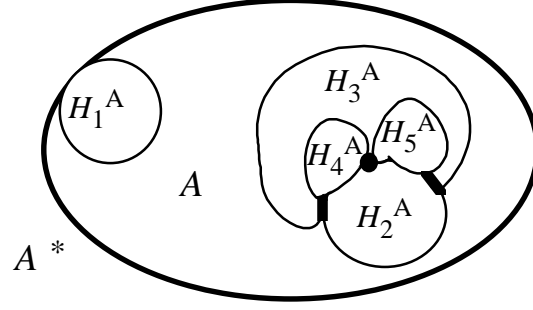


Figure 5.1: Region with holes.

A hole may have several adjacent holes. If it has more than two *meet* relations, then the hole has a *multi-meet* with respect to the other holes. This property is not necessarily symmetric since a *multi-meet* from H_0 to H_1 and H_2 does not imply the existence of a *multi-meet* for H_1 or H_2 . Figure 5.2 shows a configuration in which hole H_0 has a *multi-meet*.

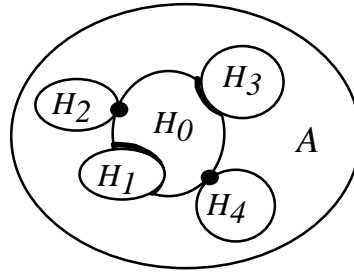


Figure 5.2: Hole H_0 has a multi-meet.

5.1.3.2 Characteristics of the Generalized Region

Since the generalized region contains each hole (i.e., $A^* \supseteq H_i^A$), the topological relation that must hold between the generalized region and each hole is *contains* or *covers* or *equal*. These properties follow immediately from the characteristics of the holes, as they

describe the relations converse to those between the hole and the generalized region (Section 5.1.3.1).

Similar to the dimension of the *meet* relation between holes, the dimension of the relation *covers* between the generalized region and a hole is based on the boundary-boundary component:

- If the dimension of all boundary-boundary components is 0, then the topological relation is called *0-dimensional covers (0-covers)*.
- If the dimension of all boundary-boundary components is 1, then the topological relation is called *1-dimensional covers (1-covers)*.
- If the dimension of some boundary-boundary components is 0, and of some others 1, then the topological relation is called *mixed-dimensional covers (mixed-covers)*.

5.1.3.3 Checking Equivalence

Paiva and Egenhofer (1995) presented an incremental algorithm to check equivalence between two regions with holes. The basic criteria to identify topological equivalence between two regions with holes are:

- the same number of holes exists in both region with holes;
- corresponding holes in both scenes have the same number and same type of *meet* intersections related with their adjacent holes and generalized region;
- corresponding holes in both scenes have the same topological relations with respect to other holes;
- corresponding holes in both scenes have the same topological relations with respect to their generalized region;

- the boundary sequence of *meet* relations among corresponding holes are equivalent; and
- the boundary sequences of *cover* relations among corresponding holes and their generalized regions are equivalent.

The region with holes corresponds to a collection of individual entities and can be seen as an individual hierarchical spatial scene. The higher level contains the generalized region and the lower level contains the holes. Therefore, the region with holes can be modeled using the relation-based model (Chapter 4). The next section introduces an algorithm to identify the topological equivalence between spatial scenes. It uses the relation-based model and the basic criteria's to identify equivalence between region with holes, as a way to build an association graph. This association graph contains possible matches between individual representations, that are used during the process of equivalence analysis.

5.2 Spatial Scenes

The process of checking the topological equivalence between two spatial scenes corresponds to a spatial matching process. It tries to find for each object representation or feature in one scene an equivalent feature in the other scene. A graph is composed of several features, and a perfect match between the features of two graphs generates an isomorphic configuration. The isomorphism occurs if and only if there is a one-to-one mapping of all representations of the two graphs such that all adjacency relationships are preserved. The association graph structure is used to identify the possible initial matches based on the topological structure of each individual representation, and then a recursive

procedure is applied to identify isomorphic configurations and the equivalent matches. The association graph is built using the relation-based model extracted from the spatial scenes. It is assumed that no semantics or any other type of information is provided, although they may be easily incorporated into the system as additional constraints. The process involves the following steps for each hierarchical level of the scene:

- Model the adjacent object representations and spatial relations between them into a graph representation, with their respective node and arc property sets (relation-based model), and build the association graph with possible matches between representations with the same characteristics.
- Identify an isomorphism between graph representations and map all possible matches for each object representation. This process is not concerned with the order in which the intersections occur between adjacent elements.
- Check the boundary sequence for each match in order to validate it.

5.2.1 Building Graph Structure and Association Graph

Figure 5.3 shows two simple scenes that are going to be used to illustrate the topological checking process. First, the equivalent relation-based model should be extracted from the scenes. Table 5.1 shows the equivalent graph nodes and boundary sequence. In this example, all *meet* relations are of type 0-dimensional.

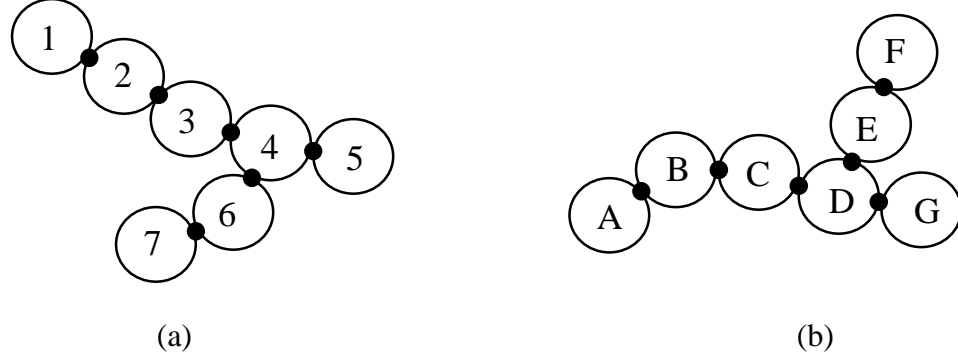


Figure 5.3: Adjacent circles representing two spatial scenes.

| Node | Def. | Sequence of the boundary components |
|-------|-------------|---|
| N_1 | { 1, area } | { (N_2 , (<i>0-meet</i> , unbounded)) } |
| N_2 | { 2, area } | { (N_1 , (<i>0-meet</i> , unbounded)), (N_3 , (<i>0-meet</i> , unbounded)) } |
| N_3 | { 3, area } | { (N_2 , (<i>0-meet</i> , unbounded)), (N_4 , (<i>0-meet</i> , unbounded)) } |
| N_4 | { 4, area } | { (N_3 , (<i>0-meet</i> , unbounded)), (N_5 , (<i>0-meet</i> , unbounded)), (N_6 , (<i>0-meet</i> , unbounded)) } |
| N_5 | { 5, area } | { (N_4 , (<i>0-meet</i> , unbounded)) } |
| N_6 | { 6, area } | { (N_4 , (<i>0-meet</i> , unbounded)), (N_7 , (<i>0-meet</i> , unbounded)) } |
| N_7 | { 7, area } | { (N_6 , (<i>0-meet</i> , unbounded)) } |
| N_A | { A, area } | { (N_B , (<i>0-meet</i> , unbounded)) } |
| N_B | { B, area } | { (N_A , (<i>0-meet</i> , unbounded)), (N_C , (<i>0-meet</i> , unbounded)) } |
| N_C | { C, area } | { (N_B , (<i>0-meet</i> , unbounded)), (N_D , (<i>0-meet</i> , unbounded)) } |
| N_D | { D, area } | { (N_C , (<i>0-meet</i> , unbounded)), (N_E , (<i>0-meet</i> , unbounded)), (N_G , (<i>0-meet</i> , unbounded)) } |
| N_E | { E, area } | { (N_D , (<i>0-meet</i> , unbounded)), (N_F , (<i>0-meet</i> , unbounded)) } |
| N_F | { F, area } | { (N_E , (<i>0-meet</i> , unbounded)) } |
| N_G | { G, area } | { (N_D , (<i>0-meet</i> , unbounded)) } |

Table 5.1: Relation-based model information for scenes of Figure 5.3.

Once the characteristics of each feature are identified in the graph model, the association graph is built. It is constructed based on the number and type of boundary components that each feature contains. Each node of the association graph corresponds to a pair of features with the same characteristics (i.e., it satisfies the constraint model), and each link in the association graph means that the features in each scene are adjacent. [Figure 5.4](#) shows the equivalent association graph for scenes of [Figure 5.3](#) in which the constraint applied is *equivalence*. For example, there is a link between nodes $\{1,A\}$ and $\{2,B\}$, because feature $\{1\}$ is adjacent to feature $\{2\}$ and feature $\{A\}$ is adjacent to feature $\{B\}$.

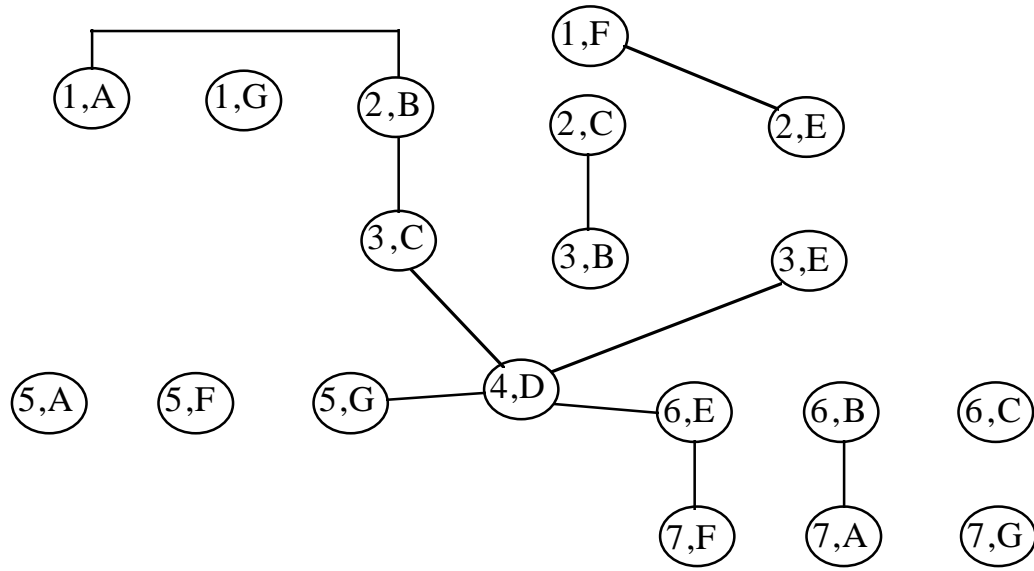


Figure 5.4: Association graph between scenes of [Figure 5.3](#), using the spatial relations.

5.2.2 Finding Isomorphic Configurations

When comparing two graphs, topologically equivalent scenes may generate one or more isomorphic configurations. A recursive procedure with backtracking, similar to the connect labeling problem (Haralick and Shapiro 1979; Haralick and Shapiro 1980)

applied to networks (Haralick et al. 1985), has been developed. Basically, for each node in the association graph, the algorithm tries to find one isomorphic configuration. The procedure starts with one node of the association graph composed by one feature of both graphs ($\text{Feat}_{s_1}, \text{Feat}_{s_2}$), which is considered the current node. Then a forward search routine is called to find if it is possible to map the adjacent elements of Feat_{s_1} onto the adjacent elements of Feat_{s_2} . The next adjacent node is assigned if it is present in the association graph, and if there is a link between it and the previous node. For each new combination found the node is taken as the current node, and the forward search routine is called again to analyze now the adjacent elements of this current node. If at some point the procedure is not able to find an equivalent situation, then it backtracks to the initial adjacent node and tries another combination between the adjacent elements. The basic search-forward procedure is described in [Algorithm 5.1](#). It has as input parameters the current node containing a pair of features, the actual current list of node matches, and the association graph used as reference.

[Table 5.2](#) shows the only isomorphic configuration that can be found for the association graph of [Figure 5.4](#). Starting with node $\{1, A\}$ the process continues to the next adjacent pair $\{2, B\}$, and so on. At node $\{4, D\}$ there are more adjacent elements to analyze, and the algorithm permutes the node combinations in order to find equivalence. First the pair $\{5, E\}$ is tested, but as it does not belong to the association graph, it is discarded and the algorithm backtracks to the next combination from node $\{4, D\}$. The next test is done on node $\{5, G\}$, which is accepted because it belongs to the association graph and it is connected with node $\{4, D\}$. As there are no more adjacent elements for features 5 and G, the algorithm backtracks to node $\{4, D\}$ and tries to match the remaining nodes. The next selection is node $\{6, E\}$, which is valid because it belongs to the

association graph and it is connected to node {4,D}. Next, node {7,F} is analyzed which is also valid, and this completes the isomorphic configuration as all features of one scene have a different match with the other scene.

```

boolean
SearchForward (Node NodePair, PairList ListOfPairs, AssociationGraph AssGraph)
Begin
    If (NodePair  $\notin$  AssGraph)
        return FALSE;
    If ListOfPairs is not null and there is no link on AssGraph
        between last node of ListOfPairs and NodePair
        return FALSE;
    Add NodePair to ListOfPairs;
    Get adj. boundaries of NodePairs1 (first scene), and NodePairs2 (second scene)
    For each adjacent boundary Adjs1 of NodePairs1 that is not in ListOfPairs
        For each adj. boundary Adjs2 of NodePairs2 that is not in ListOfPairs
            NewNode  $\leftarrow$  (Adjs1, Adjs2);
            If (SearchForward (NewNode, ListOfPairs, AssGraph))
                Set flag that Adjs1, and Adjs2 are in ListOfPairs;
    If all adjacent boundaries of NodePairs1, and NodePairs2 are in ListOfPairs
        return TRUE;
    Else
        Remove NodePair from ListOfPairs;
    return FALSE;
End

```

Algorithm 5.1: Recursive procedure to identify isomorphic configurations.

| |
|--|
| $\{1,A\} \rightarrow \{2,B\} \rightarrow \{3,C\} \rightarrow \{4,D\} \rightarrow \{5,E\}$ failed, then backtracks $\{4,D\} \rightarrow \{5,G\}$ ok, then backtracks $\{4,D\} \rightarrow \{6,E\} \rightarrow \{7,F\}$ then ends. |
|--|

Table 5.2: Isomorphic configuration process based on [Figure 5.4](#).

5.2.3 Validating Isomorphic Configurations

Topologically equivalent scenes generate isomorphic configurations, but the fact that two configurations are isomorphic does not mean that the configurations are topologically equivalent. In order to guarantee that the scenes or configurations are equivalent in terms of topology, one last test, checking the boundary sequences for the features in each node, needs to be performed. The boundary sequences are shown in [Table 5.1](#). Each boundary component has the feature identifier, the relation type, and the complement relationship value. The boundary sequences of a matching pair i,j will be equivalent if:

- they have the same number of boundary elements n ; and
- for each boundary component k ($1 \leq k \leq n$) of i and j the feature identifiers corresponds to a node in the isomorphic configuration, and the boundary type and complement relationship values are equal. The boundary sequence of the feature in one scene is kept constant, while the other boundary sequence for the feature of the second scene is clockwise permuted until an equivalence is found, or all permutations are analyzed.

The combination between the isomorphic configuration of [Table 5.2](#) and the boundary sequences of [Table 5.1](#) generates [Table 5.3](#), which identifies valid matches after analyzing the boundary sequences. For simplification the boundary types are not

represented in this table, as they are all of the same type *0-meet, unbounded*. If all matches are valid, then the two spatial configurations have the same topology. [Table 5.3](#) shows that the boundary sequences for node (4,D) is invalid, because for each possible combination there is at least one pair of feature identifiers that does not belong to the isomorphic configuration found.

| Node | Sequence of boundary components |
|-------|--|
| (1,A) | { (2) (B) - valid } |
| (2,B) | { (1,3) (A,C) - valid } { (1,3) (C,A) - invalid } |
| (3,C) | { (2,4) (B,D) - valid } { (2,4) (D,B) - invalid } |
| (4,D) | { (3,5,6) (C,E,G) - invalid } { (3,5,6) (G,C,E) - invalid } { (3,5,6) (E,G,C) - invalid } |
| (5,G) | { (4,D) - valid } |
| (6,E) | { (4,7) (D,F) - valid } { (4,7) (F,G) - invalid } |
| (7,F) | { (6,E) - valid } |

Table 5.3: Boundary components sequence for nodes of [Table 5.2](#)

5.2.4 General Procedure

The procedure to identify isomorphism ([Algorithm 5.1](#)) uses the association graph between two graphs of spatial scenes. A spatial scene may have several graphs, at different levels of detail. Therefore, when comparing two spatial scenes, the isomorphism process should be applied between graphs at the same level of hierarchy, starting at the higher levels and going towards the lower levels. At lower levels of the hierarchy, the isomorphism check is performed just between graphs in which the parent graphs correspond to a node in one isomorphic configuration. A collection of isomorphic

configurations represents a global match between two scenes, and several collections may be found ([Algorithm 5.2](#)).

```

CollIsoMConfList  EvaluateTopology (SpatialScene  $S_1$ , SpatialScene  $S_2$ )
Begin
    Make collection list of isomorphic configurations CollIsoList equal null;
    For each level  $L$  of Scene  $S_1$ :
        Begin
            Make a temporary list of isomorphic configurations TempIsoL equal null;
            Get graphs of  $S_1$  and  $S_2$  at level  $L$  and put them into graph lists Glist1 and Glist2;
            For each combination of graphs  $G_1$  and  $G_2$  from Glist1 and Glist2:
                Begin
                    If same number of features between  $G_1$  and  $G_2$ 
                        Begin
                            If level  $L > 1$ 
                                If father graphs of  $G_1$  and  $G_2$  are not part of CollIso then
                                    go to next combination of  $G_1$  and  $G_2$ ;
                                Build an association graph AssGr with  $G_1$  and  $G_2$ ;
                                Find the isomorphic configurations in AssGr and add them
                                    to TempIsoL;
                            End
                        End
                    Add the contents of temporary list TempIsoL into the current collection list of
                    isomorphic configurations CollIsoList, avoiding repetition of graph components.
                    Look for all possible combinations;
                End
            Return the collection list of isomorphic configurations CollIsoList;
        End
    End

```

Algorithm 5.2: Procedure to compare topology between two scenes.

[Algorithm 5.2](#) describes the process to compare topological aspects between different representations. It is based on the relation-based model that contains all the necessary topological information. There is no need for deriving any other information. Representation models based on pure geometry would require the derivation of all topological relationships before using [Algorithm 5.2](#). For the cell complex representation model, which contains geometry and some topological information, it would be also necessary additional processing in order to derive topological relationships related to 0-dimensional adjacencies. Therefore, the development of topological tools based on the relation-based model simplifies the analysis of comparing different spatial representations, and supports the hypothesis of this thesis.

5.3 Similarity Measures

The previous section presented an algorithm to determine equivalence between two spatial scenes by identifying the matching of features in both configurations. Spatial scenes may be affected by some transformations that may change their general topological structure. With some changes it is possible to consider that both scenes are similar. Similarity is the assessment of deviation from equivalence (Bruns and Egenhofer 1996). Using some similarity constraints based on comparison of number of intersections, dimension of intersections, and changes in spatial relations, we are able to relax the equivalence model and use the same approach as for identifying the isomorphic configurations. This section describes similarity measures for individual features in spatial scenes, similarity measures for the general structure of a spatial scene, and similarity measures for isomorphic configurations between two spatial scenes.

5.3.1 Individual Representation

Each spatial object may be described by one or more spatial representations. These representations are nodes on the relation-based model. Each node on the graph may contain links that relate to the adjacent representations, as well to the hierarchy levels of the spatial scene. Some similarity measures can be extracted for each individual representation considering its individual characteristics and its adjacent features.

5.3.1.1 Dimension

The representation dimension may vary from a 2-dimensional region, to a 1-dimensional line, to a 0-dimensional point. The dimension similarity value between two representations i, j may be obtained by using the absolute difference of the representation dimensions divided by 2 (which is the maximum possible difference). Equation 5.2 represents the similarity value in which values close to 1 mean more similarity.

$$DimensionSim_{i,j} = 1 - \left(\frac{abs(dimension_i - dimension_j)}{2} \right) \quad (5.2)$$

5.3.1.2 Number of Adjacent Elements

The graph is modeled using the spatial relations between the scene's object representations or features. Adjacent elements are related by 0- or 1-dimensional meets, and a similarity measure between two individual graph features can be obtained by computing the ratio between minimum and maximum values of 0-dimensional (Equation 5.3) and 1-dimensional *meets* (Equation 5.4) and getting the average (Equation 5.5).

$$0 - meetSim_{i,j} = \frac{\min(\#0 - meets_i, \#0 - meets_j)}{\max(\#0 - meets_i, \#0 - meets_j)} \quad (5.3)$$

$$1 - meetSim_{i,j} = \frac{\min(\#1 - meets_i, \#1 - meets_j)}{\max(\#1 - meets_i, \#1 - meets_j)} \quad (5.4)$$

$$MeetSim_{i,j} = \frac{0 - meetSim_{i,j} + 1 - meetSim_{i,j}}{2} \quad (5.5)$$

Figure 5.5 shows several regions with a selected feature and its adjacent elements. For this selected feature, in Figure 5.5a the total number of 0-dimensional *meets* is zero, and the total number of 1-dimensional *meets* is 5, while in Figure 5.5b two features have been merged and the number of 1-dimensional *meets* is now 4. Therefore, the similarity measure between the selected features for the number of adjacent elements, based on

Equations 5.3, 5.4, 5.5 is: $\frac{1 + \frac{4}{5}}{2} = 0.9$.

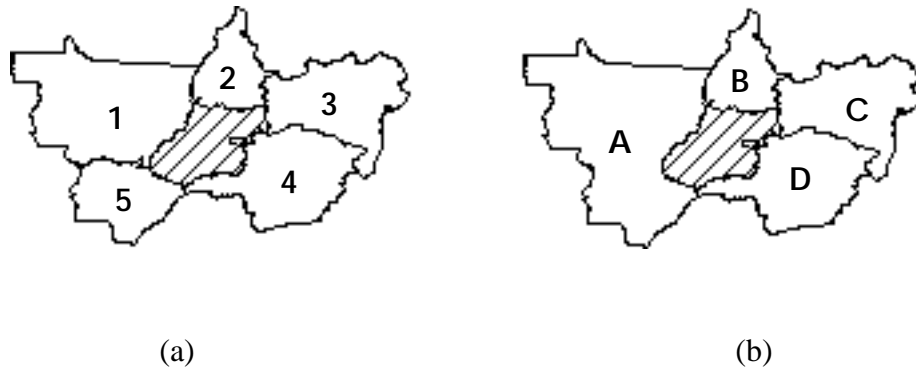


Figure 5.5: Selected feature and its adjacent features.

5.3.1.3 Adjacent Structure

The adjacent structure of a feature can be described by the spatial relations between its adjacent elements. These spatial relations may be *meet* or *disjoint*, and the measure for the adjacent structure is defined as the number of *meet* relations between the adjacent elements. For example, in [Figure 5.5a](#) there are five *meet* relations between the adjacent elements of the selected feature $\{(1,2), (1,5), (2,3), (3,4), (4,5)\}$, while for [Figure 5.5b](#) there are four *meet* relations $\{(A,B), (A,D), (B,C), (C,D)\}$. The adjacent structure similarity measure between two features i, j is measured by [Equation 5.6](#).

$$AdjacentStructureSim_{i,j} = \frac{\min(\#adjacentmeet_i, \#adjacentmeet_j)}{\max(\#adjacentmeet_i, \#adjacentmeet_j)} \quad (5.6)$$

5.3.1.4 Hierarchical Structure

The hierarchical structure of an area feature can be measured by the number of internal graphs that it contains. Each graph on the relation-based model represents a set of connected features or an isolated feature. Therefore, the basic measure for the hierarchical structure of an area feature A is equal to the number of graphs for which the parent feature is A . This measure for the number of lower levels graphs is called the number of *childgraphs*. [Equation 5.7](#) represents the hierarchical structure similarity between two features.

$$HierarchicalStructureSim_{i,j} = \frac{\min(\#childgraphs_i, \#childgraphs_j)}{\max(\#childgraphs_i, \#childgraphs_j)} \quad (5.7)$$

In [Figures 5.5a and 5.5b](#) the selected features have one lower level graph representing the internal region, therefore, the similarity value for the hierarchical structure is 1 for these selected features.

5.3.2 Spatial Scenes

The previous measurements were concerned about individual features. A spatial scene may contain several objects with many representations, and it is possible to extract some similarity values in terms of the general structure of the scene, referred to as *detailed similarity*, as well as measures for isomorphic configurations between different scenes, referred to as *topological similarity*.

5.3.2.1 Detailed Similarity

Detailed similarity measures refer to the general structure of the spatial scene. Two detailed similarity values may be obtained:

- the number of hierarchical levels on the scene; and
- the number of graphs per level of hierarchy.

The detailed similarity between two scenes i, j considering the number of levels may be computed with [Equation 5.8](#), which calculates the ratio between the number of levels in both scenes.

$$HierarchicalSim_{i,j} = \frac{\min(\#levels_i, \#levels_j)}{\max(\#levels_i, \#levels_j)} \quad (5.8)$$

The similarity based on the number of graphs per level of hierarchy can be obtained by computing the similarity between each level, and then dividing the total by the maximum number of levels between scenes i, j . Equation 5.9 computes the similarity between each level l with the ratio between the number of graphs at level l , and Equation 5.10 computes the total similarity between the scenes, assuming l as the maximum level. These equations are very simple, and they do not capture how the lower level graphs are related to their parent graphs, but they give an idea of the graph structure. Several different methods to compare graph structures can be found in standard graph theory (Balakrishnan 1997).

$$LevelSim_{(i,j)_l} = \frac{\min(\#graphs_{i,l}, \#graphs_{j,l})}{\max(\#graphs_{i,l}, \#graphs_{j,l})} \quad (5.9)$$

$$GraphLevelSim_{i,j} = \frac{\sum_{l=1}^l LevSim_{(i,j)_l}}{l} \quad (5.10)$$

5.3.2.2 Topological Similarity

The topological similarity between two scenes applies to isomorphic configurations. Each isomorphic configuration represents a match of features between two graphs. This match corresponds to a 1:1 mapping between features of both graphs. For scenes with n graphs ($n > 1$), there may exist a collection of isomorphic configurations describing the matching of features between the graphs. Each collection of isomorphic configurations describes one possible matching of features between the two scenes. Several collections of isomorphic configurations may be found when comparing the spatial scenes. The

following measurements can be extracted from each collection of isomorphic configurations:

- using the nodes of the isomorphic configurations, compute the number of valid matching, where valid means that the boundary sequences between the node features satisfy the constraints applied;
- using the nodes of the isomorphic configurations, compute the dimension similarity between the node features;
- using the nodes of the isomorphic configurations, compute the spatial relation similarity between the node features; and
- using the number of features, the number of 0-dimensional *meets* and the number of 1-dimensional *meets* that each graph contains, compute the graph structure similarity between the graphs of the isomorphic configurations.

5.3.2.2.1 Valid Matching

Each isomorphic configuration is characterized by a set of nodes, with each node containing a feature of one scene and the equivalent matching feature on the other scene. The matching is considered valid if the boundary sequence of both features follow the similarity constraints. In cases of equivalence, the boundary sequences must be the same.

Each isomorphic node may be defined by a pair of features and a flag identifying if the boundary sequence is valid: $IsoNode = \{(F_{S_1}, F_{S_2}) IsValid\}$. Each isomorphic

configuration contains a collection of nodes, $IsoConf = \sum_{i \rightarrow 1}^n IsoNode_i$, where n is the

number of nodes. An isomorphic collection is a list of isomorphic configurations,

$IsoColl = \sum_{i \rightarrow 1}^m IsoConf_i$, where m is the number of isomorphic configurations. The

similarity value for valid matching on an isomorphic configuration is the number of valid matching divided by the number of nodes ([Equation 5.11](#)).

$$ValidSim_{IsoConf} = \frac{\sum_{i=1}^n k \ (k = 1 \text{ if } IsoNode \rightarrow IsValid, \text{ otherwise } k = 0)}{n} \quad (5.11)$$

Using [Equation 5.11](#), the valid similarity for each collection of isomorphic configurations is the average of all valid similarity over m isomorphic configurations ([Equation 5.12](#)).

$$ValidSim_{IsoColl} = \frac{\sum_{i=1}^m ValidSim_{IsoConf_i}}{m} \quad (5.12)$$

5.3.2.2.2 Feature Dimensions

The dimension similarity measure for each isomorphic configuration is determined by computing the dimension similarity between features of each node, divided by the number of nodes, n ([Equation 5.13](#)).

$$DimSim_{IsoConf} = \frac{\sum_{k=1}^n DimSim_{IsoConf \rightarrow N_k}}{n} \quad (5.13)$$

Using [Equation 5.13](#), the dimension similarity for each collection of isomorphic configurations is the average of the dimension similarity of m isomorphic configurations ([Equation 5.14](#)).

$$DimSim_{IsoColl} = \frac{\sum_{i \rightarrow 1}^m DimSim_{IsoConf_i}}{m} \quad (5.14)$$

5.3.2.2.3 Spatial Relation between Features

Topological changes may occur for individual objects, such as dropping parts or reducing dimension, but they may also occur with the topological spatial relations between the objects. Topological relations capture the characteristics of the spatial configurations, and gradual changes in topology may cause two equivalent scenes to become less similar (Bruns and Egenhofer 1996). The conceptual neighborhood of topological relations has been modeled with the concept of gradual changes (Egenhofer and Al-Taha 1992), and it supports the determination of similar relations by using an ordering scheme for topological relations. Bruns and Egenhofer (1996) described the conceptual neighbors of topological and direction relations between regions, and Egenhofer and Mark (1995) modeled the conceptual neighborhoods of topological line-region relations. [Table 5.4](#) shows the conceptual neighborhood distances for the content invariants of spatial relations between two simple regions. For example, two *disjoint* regions turn into an *inside* or *contains* relation if four units of gradual changes occur.

| | <i>Disjoint</i> | <i>Meet</i> | <i>Overlap</i> | <i>Covers</i> | <i>Covered By</i> | <i>Contain</i> | <i>Inside</i> | <i>Equal</i> |
|-------------------|-----------------|-------------|----------------|---------------|-------------------|----------------|---------------|--------------|
| <i>Disjoint</i> | 0 | 1 | 2 | 3 | 3 | 4 | 4 | 3 |
| <i>Meet</i> | | 0 | 1 | 2 | 2 | 3 | 3 | 2 |
| <i>Overlap</i> | | | 0 | 1 | 1 | 2 | 2 | 1 |
| <i>Covers</i> | | | | 0 | 2 | 2 | 2 | 1 |
| <i>Covered By</i> | | | | | 0 | 2 | 1 | 1 |
| <i>Contain</i> | | | | | | 0 | 2 | 1 |
| <i>Inside</i> | | | | | | | 0 | 1 |
| <i>Equal</i> | | | | | | | | 0 |

Table 5.4: Difference matrix (*DiffTopo*) for the conceptual neighbors between regions.

Given two regions A and B that exist in both scenes S_1 and S_2 , the topological similarity for the spatial relation between A and B considering both scenes is expressed as the ratio between the distance of the spatial relations and the maximum distance i.e., 4, [Equation 5.15](#).

$$TopologicalSim_{A,B} = 1 - \frac{DiffTopo[SR_{A,B}^{S_1}, SR_{A,B}^{S_2}]}{4} \quad (5.15)$$

For an isomorphic configuration the topological similarity measure is calculated by taking the equivalent pair of features in both scenes, and then computing the topological similarity using [Equation 5.15](#). The total similarity is the sum of the measure for each feature pair, divided by the number of feature pairs.

5.3.2.2.4 Graph Structure

The previous three similarity measures took into consideration the pair of features found on the isomorphic configurations, but they did not account for features of graphs that may not be matched, or for features that may be merged, split, or dropped. These kind of situations may change the graph structure. Given a graph, three measures can be extracted from it:

- the number of 0-dimensional meets;
- the number of 1-dimensional meets; and
- the number of features.

The similarity measure for the graph structure between two graphs i, j can be computed adding the ratio between minimum and maximum values for the above parameters (Equations 5.16, 5.17, 5.18). This similarity measure is represented on Equation 5.19.

$$Graph0-meetSim_{i,j} = \frac{\min(\#0-meets_i, \#0-meets_j)}{\max(\#0-meets_i, \#0-meets_j)} \quad (5.16)$$

$$Graph1-meetSim_{i,j} = \frac{\min(\#1-meets_i, \#1-meets_j)}{\max(\#1-meets_i, \#1-meets_j)} \quad (5.17)$$

$$GraphFeatSim_{i,j} = \frac{\min(\#features_i, \#features_j)}{\max(\#features_i, \#features_j)} \quad (5.18)$$

$$GraphSim_{i,j} = \frac{Graph0-meetSim_{i,j} + Graph1-meetSim_{i,j} + GraphFeatSim_{i,j}}{3} \quad (5.19)$$

Given a collection of isomorphic configurations representing a match between two scenes, the graph similarity measure for one of the scenes S_1 is defined as the sum of the graph similarity for each isomorphic configuration that is part of the collection, divided by the number of graphs in scene S_1 (Equation 5.20).

$$GraphSim_{IsoColl, S_1} = \frac{\sum_{i=1}^n GraphSim_{IsoGraph_i \rightarrow G_1, IsoGraph_i \rightarrow G_2}}{\# graphs_{S_1}} \quad (5.20)$$

5.4 Topological Changes

The proposed procedure to identify isomorphism (Algorithm 5.2) compares graphs with the same number of features. However, topological changes such as merging, splitting, or dropping may change the number of features in one configuration. In order to use this algorithm, in cases of merging for example, it is necessary first to identify possible merges of features, second to perform a merge operation on the relation-based model, and third to apply the procedure for checking isomorphism (Algorithm 5.1). The identification of possible merges can be done by using the similarity measures presented in the previous section.

5.4.1 Merging

Given two graphs G_1 and G_2 with m and n features respectively ($m > n$), the problem consists in identifying pair of features in G_1 that may correspond to one feature in G_2 . A merge operation on the relation-based model corresponds to eliminating the boundary components between the features to be merged. After merging features, the structure of the adjacent elements for the merged feature is modified in terms of the total number of adjacent elements, and how they are related. The similarity values for individual features

(Equations 5.2, 5.5, 5.6, and 5.7) are used to verify if a pair of features in G_1 feature are candidate to be a feature in G_2 . Figure 5.6 shows a merge example with two situations for the political subdivision of Brazil, in which Figure 5.6a contains one more feature than Figure 5.6b.

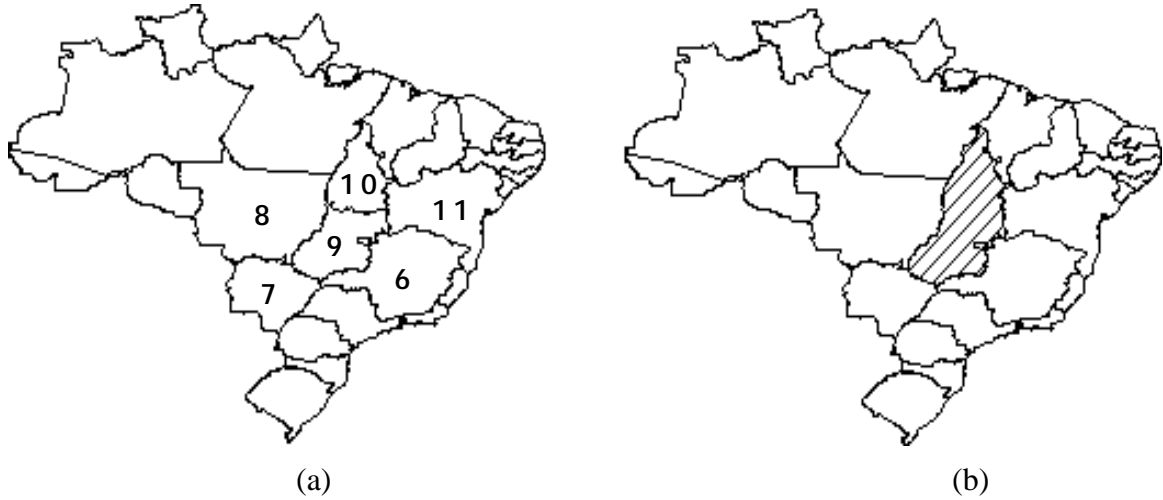


Figure 5.6: Political States of Brazil.

The procedure to identify possible pair of features that may result in a union is:

- Select a feature in the graph with less features (G_2).
- Look for similar features in graph G_1 using the similarity values for individual representations: dimension similarity (Equation 5.2), number of *meets* similarity (Equation 5.5), adjacent structure similarity (Equation 5.6), and hierarchical similarity (Equation 5.7).
- For each similar feature that satisfies a tolerance value, calculate how similar is the merge of this feature with one of its adjacent elements, compared with

the selected feature. This is done by using the similarity values for the number of *meets* (Equation 5.5) and for the adjacent structure (Equation 5.6).

- Merge with high values of similarity are performed on the graph model and the isomorphism procedure (Algorithm 5.2) is executed.

The following examines the example for the selected feature of Figure 5.6b. Table 5.5 shows the features in Figure 5.6a that are at least 85% similar to the selected feature. The total similarity value is the average of the four or three (not considering hierarchy similarity) similarity values between individual representations.

| Feature | Similarity | | | | |
|---------|------------|--------|----------|-----------|--------|
| | Dimension | Meet | Adjacent | Hierarchy | Total |
| 9 | 1 | 0.7143 | 0.7143 | 1 | 0.8571 |
| 8 | 1 | 0.8571 | 0.7143 | - | 0.8571 |
| 6 | 1 | 0.8571 | 0.8571 | - | 0.9048 |
| 10 | 1 | 0.8571 | 0.8571 | - | 0.9048 |
| 11 | 1 | 0.8750 | 1 | - | 0.9583 |

Table 5.5: Similarity values between features of Figure 5.6a and selected feature in Figure 5.6b.

With the identification of the similar features, the next step corresponds to evaluate the *meet* similarity and adjacent structure similarity between the selected feature and each possible merge for the similar features. For each similar feature of Table 5.5, perform a merge with each of its adjacent features and compare it with the selected feature. Figure 5.7a displays the adjacent features for the merging between features 9 and 7, and Figure 5.7b highlights the adjacent features for the selected element.

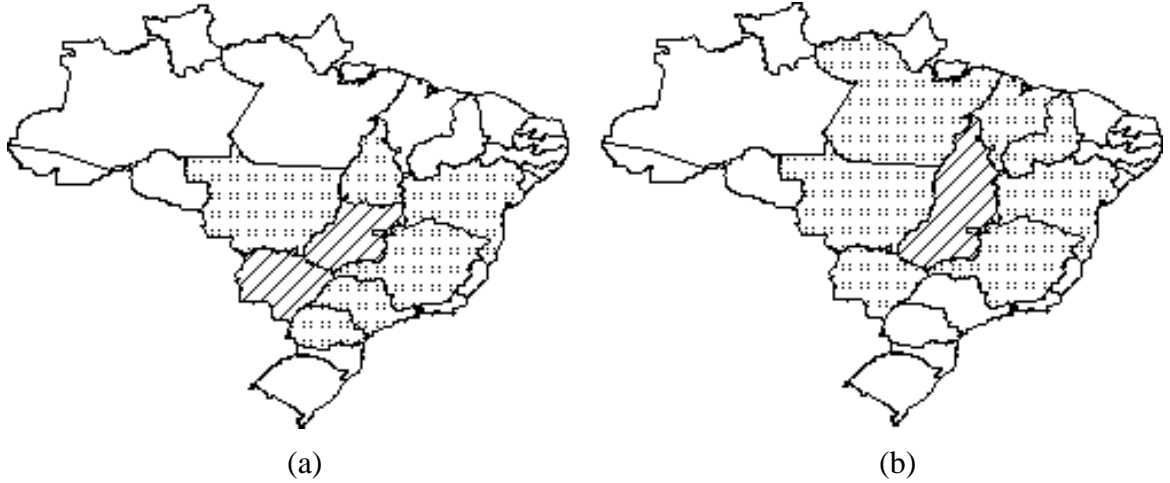


Figure 5.7: Adjacent features for: (a) merged features; (b) individual feature.

Table 5.6 shows the similarity values when merging feature 9 of Figure 5.6a with each of its adjacent features and comparing it with the selected feature in Figure 5.6b. Similarity values close to 1 should be taken into consideration, and the merge operation is performed in the graph before the analysis of equivalence. One important point is to identify which is the similarity threshold value to be used as reference in this case. When the compared graphs have just one feature of difference, it is expected that the merged features have the same structure of the selected feature, and in this case the similarity threshold value would be 1. However, as the difference in terms of number of features between the graphs increases it is possible that adjacent merges occur and in this case the adjacent similarity value for comparison may not be 1 but a little less.

The relation-based model should provide a merge operation. The merge procedure between two area features on the graph model is detailed in Algorithm 5.3. The approach taken to analyze merge situations can be used in situations of splitting, by making the analysis with the scenes in the reverse order.

| Merge | #Meets | Adj. Structure | Similarity | | |
|--------|--------|-------------------|------------|----------|--------|
| | | | Meet | Adjacent | Total |
| {9,10} | 7 | 7 | 1 | 1 | 1 |
| {9,6} | 7 | 7 | 1 | 1 | 1 |
| {9,8} | 7 | 6 | 1 | 0.857 | 0.9375 |
| {9,11} | 9 | 8 | 0.7777 | 0.875 | 0.8263 |
| {9,7} | 6 | 5 | 0.8571 | 0.7143 | 0.7857 |

Table 5.6: Meet and adjacent structure similarity values between merge of features from [Figure 5.6a](#) and selected feature in [Figure 5.6b](#).

AreaFeature* SpatialScene::Merge(**AreaFeature** **first*, **AreaFeature** **second*)

Begin

Create new area feature *newAreaFeature*;
Generate boundary list for *newAreaFeature* excluding the common boundaries between *first* and *second* features;
Update boundary list of adjacent and cover features by excluding the boundary components related with *first* and *second*, and by adding the boundaries with *newAreaFeature*;
Update graph hierarchy if necessary (if *first* and *second* have more than one boundary in common);
Remove *first* from its graph *G*;
Remove *second* from its graph *G*;
Add *newAreaFeature* to graph *G*;
return *newAreaFeature*;

End

Algorithm 5.3: Merge operation between two area features in the graph model.

5.4.2 Dropping

The operation drop corresponds to the elimination of some isolated feature from a spatial scene. It results in one graph less on the resultant scene, altering the hierarchical configuration. Therefore, when comparing scenes where dropping of features has occurred, there will be no match for some graphs, which will result in a less similar measure for graph structure between the collections of isomorphic configurations.

[Algorithm 5.4](#) shows the procedure for dropping an isolated feature in the graph model.

```
boolean SpatialScene::Drop(Feature *feature)  
Begin  
    If feature type is equal to AREA then update graph hierarchy if necessary;  
    Remove feature from its graph G;  
    return true;  
End
```

Algorithm 5.4: Drop operation in the graph model.

5.5 Summary

A recursive procedure to identify equivalence between spatial scenes has been presented. It is based on the information extracted from the relation-based model that describes a spatial scene. The association graph is created using the spatial relations between the graph features. To analyze equivalence, the association graph nodes should contain a pair of features with the same characteristics. However, the association graph can be built using some degree of similarity between features, which allows for similarity analysis between spatial scenes. A set of similarity measures between individual representations, between scene structures, and between isomorphic configurations can be used as a way to

relax the concept of equivalent scenes into the concept of similar scenes. The measures presented do not use weights, however weights can be assigned depending on the type of analysis. For example, when analyzing scenes at the same level, dimension and adjacent parameters may be more important, while for scenes at different levels the hierarchical structure may be an important parameter to consider. The information captured by the relation-based model can be stored in the database as the metadata description for the spatial relations between object representations and the similarity measures between individual representations can be used as a search procedure in digital libraries.

Chapter 6

Software Implementation

This chapter describes an object model that supports the implementation of a topological consistency checker that is based on the relation-based model (Chapter 4). Object orientation is a software modeling methodology that facilitates the design and construction of complex systems from individual components. It provides concepts and tools that permit developers to model and represent the real world as closely as possible. The object-oriented approach is characterized by objects and abstraction mechanisms to deal with them. Each data object contains operations that describe its behavior. Groups of data objects that have the same operations are implemented through classes. A class describes and implements all the operations to manipulate its instances. Abstraction tools such as classification, generalization, association, and aggregation, are basic concepts for the design of object-oriented models (Brodie 1984). The abstraction concept of classification corresponds to mapping an object onto a common class. Generalization groups several classes of objects with common operations and properties into a more general class. These abstraction tools, combined with the concepts of inheritance and propagation, permit us to model complex spatial objects and represent them at different abstraction levels better than the relational model (Egenhofer and Frank 1989). The

concept of inheritance means that the properties and operations of a parent class are inherited by all related children classes. The inheritance is simple when the child class has just one parent class. If the child class has more than one parent class then the inheritance is called multiple. Some of the major benefits of the object-oriented approach is that the software components can be easily reused, modified, and extended.

Object-oriented design methods help developers to exploit the expressive power of object-based and object-oriented programming languages, using the classes and objects as basic building blocks (Booch 1994). The object model has been influenced by object-oriented programming languages, and the object-oriented analysis and design represents an evolution for the development of systems. Object-oriented analysis identifies the system requirements in terms of objects and classes, and this result serves as a model for the object-oriented design process. Object-oriented design leads to an object-oriented decomposition and uses different notations to express the different models of the class and object structure. This chapter uses the Unified Modeling Language (UML) (Booch et al. 1996) as a software engineering tool that supports the development of the topological consistency checker classes. UML is a third-generation method for specifying, visualizing, and documenting the components of an object-oriented system. It represents the unification of Booch (Booch 1994), Objectory (Jacobson 1992), and OMT (Rumbaugh 1991) methods.

This chapter is organized as follow: the first part introduces the UML notation and describes the classes and relationships for the relation-based model, and the second part describes how to convert SPRING's (INPE/DPI 1997) vector model, which is based on the cell complex structure, into the qualitative model presented in this thesis. SPRING is

a GIS and remote sensing image processing system that integrates raster and vector data, using the object-oriented approach.

6.1 UML Notation

UML distinguishes between the notions of model and diagram. A model contains all the elements of the system, and the diagram is a particular visualization of certain types of elements from a model, exposing in some cases detailed information. There are several diagram types in the UML definition, but this text uses only the class diagram and describes how to represent relationships between several class diagrams.

6.1.1 Class Diagrams

The class diagram is the core for a UML model, and it shows the important abstractions in the system and how they relate to each other. The basic elements found in class diagrams are class icons and relationship icons. UML represents individual classes as solid rectangles that may be divided into three parts or compartments. The first part contains the name of the class. The second and third parts are optional and may be used to list the attributes and operations of the class. [Figure 6.1](#) shows the class diagram for the class Point.

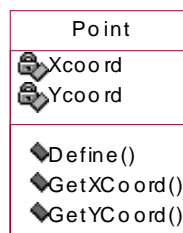


Figure 6.1: Class icons with attributes and operations.

6.1.2 Relationships

Besides the individual classes with their attributes and operations, class diagrams also represent the relationships that exist between dependent classes. UML identifies several types of relationships with their respective graphic representations. An association between two classes is depicted by connecting the classes with a straight line. The values of role-1 and role-2 specify how many instances are to participate in an association. Associations are bi-directional by default (Figure 6.2a), but UML uses an arrowhead to represent an unidirectional association (Figure 6.2b). Aggregation is a special form of association that is used to show that one object is at least partially composed of another. Figure 6.2c shows an aggregation with a hollow diamond, which represents that the whole object maintains a pointer or a reference to its parts. If the diamond is filled (Figure 6.2d), then the diagram shows that the aggregation is by value, i.e., the whole object declares an actual instance of the part object within itself. When one class shares the structure and behavior defined by another class, a diagram as in Figure 6.2e is used, showing that the subclass inherits all the attributes and operations of the superclass.

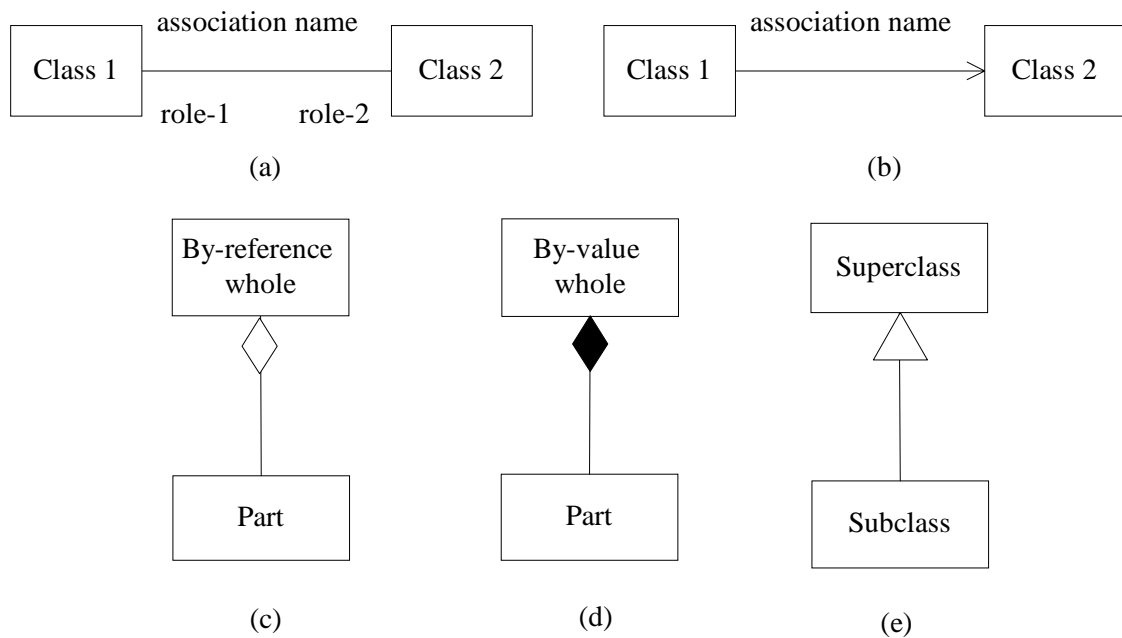


Figure 6.2: Types of UML relationships: (a) bi-directional association; (b) unidirectional association; (c) aggregation by reference; (d) aggregation by value; and (e) inheritance.

6.2 Relation-Based Model Class Structure

The relation-based model represents a spatial scene as a collection of hierarchical graphs.

There are four basic classes:

- A feature describes the object representations. Derivations from this class are point, linear, and area features.
- A boundary describes the adjacent intersection between two features.
- A graph describes a collection of connected features or isolated features.
- A spatial scene describes all object representations and relationships of one map.

Figure 6.3 shows the UML diagram that describes the relation-based model. The spatial scene class *SpatialScene* is composed by a list of graphs, the class *GraphList*. This list of graphs may be empty or not, and there is an aggregation between class *GraphList* with class *Graph*. Each graph represents a set of connected features or an isolated feature. In case of connected features, the boundary components between these adjacent features are part of the graph class. The diagram shows that the class *Graph* contains one list of features, *FeatureList* class, and one list of boundaries, *BoundaryList* class. In order to represent the hierarchical structure of the spatial scene, each graph has pointers to its parent graph and parent feature, which will be null if the graph is on the higher level of the hierarchy. Each list of features, *FeatureList* class, may have 0 or n features. Likewise each list of boundaries, *BoundaryList* class, may have 0 or m boundaries. The class *Feature* is the generalized representation for the spatial objects. The subclasses *PointFeature*, *LinearFeature*, and *AreaFeature* are derived from the class *Feature* and inherit all the attributes and operations of this class. The area feature is a special case, because it may contain or cover other graphs. The diagram shows that the *AreaFeature* class contains a pointer to the lower-level graphs that it contains, and it also contains an extra list of boundaries that describes the *covers* relations with the lower-level features.

In order to handle a list of instances of a class, all basic classes may be derived from a general class that is manipulated by a generic list. Implementations of double-linked list can be found in text books of data structures(e.g., (Knuth 1968)). Figure 6.4 shows the class diagrams for a double-linked list implemented in SPRING (INPE/DPI 1997), in which the class *SObject* represents a generic object that is part of a node of the list. Each node points to the next and to the previous node, and the list class has two pointers to nodes that represents the root and the current items of the list.

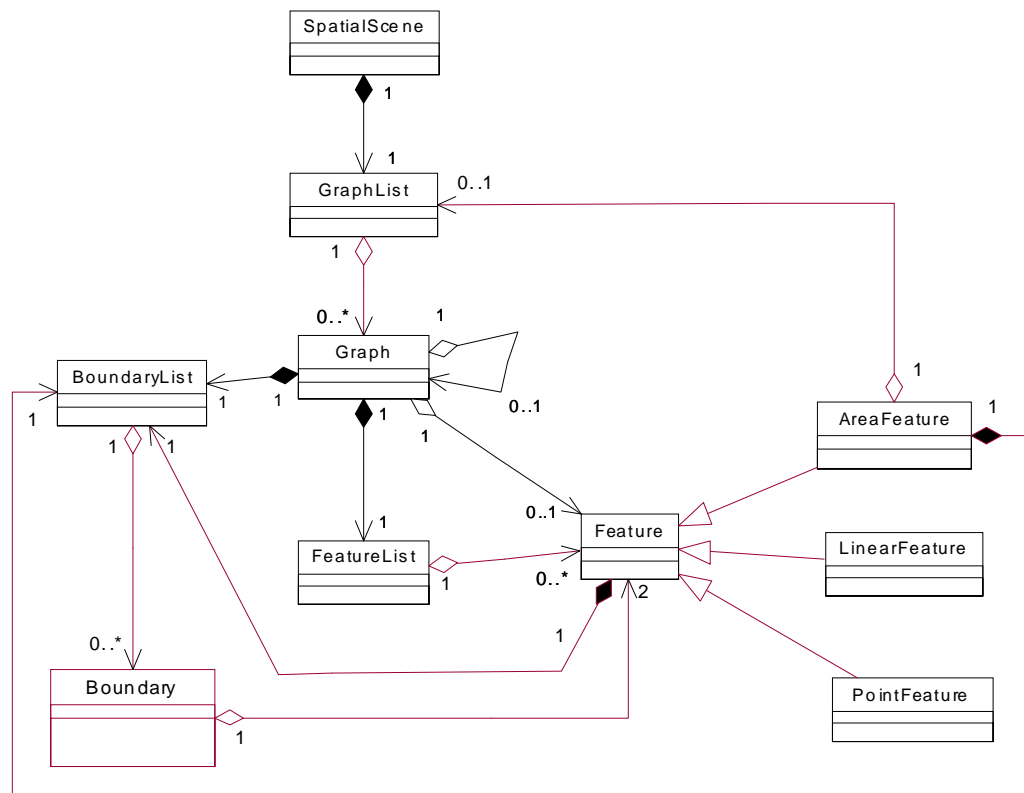


Figure 6.3: Class hierarchy for the relation-based model.

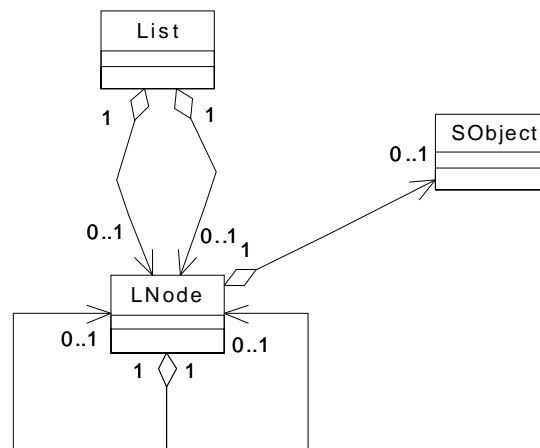


Figure 6.4: List and its node class structure.

6.2.1 *Feature*

The class *Feature* is the core of the definition of the spatial scene. It describes the representation types that the spatial objects may have. In an environment with multiple representations, one spatial object may be represented by geometric primitives or symbols describing these primitives with different dimensions. [Figure 6.5](#) shows the class diagrams for the *Feature* class and its derivation classes. The basic class *Feature* contains three attributes, a boundary list, and a set of operations that are inherited by its children's classes. The attributes are the feature identifier, the feature representation, and the geometry identifier. Although the relation-based model uses symbolic geometry, the geometry identifier attribute is provided for situations where the geometry is available and it is desirable to link the symbolic information with the geometry. The derived classes represent the feature with different dimensions. The *AreaFeature* is a special case, because it may contain other features inside it, or it may cover a set of features. Therefore, it provides an additional boundary list that stores the boundary components related to *covers* relations, and it also contains a pointer to a list of graphs that are immediately under it in the hierarchy structure.

The complete specifications for these feature classes, as well for the future classes to be presented in this chapter, are described in Appendix A. The specifications describe in detail the attributes and operations for each class.

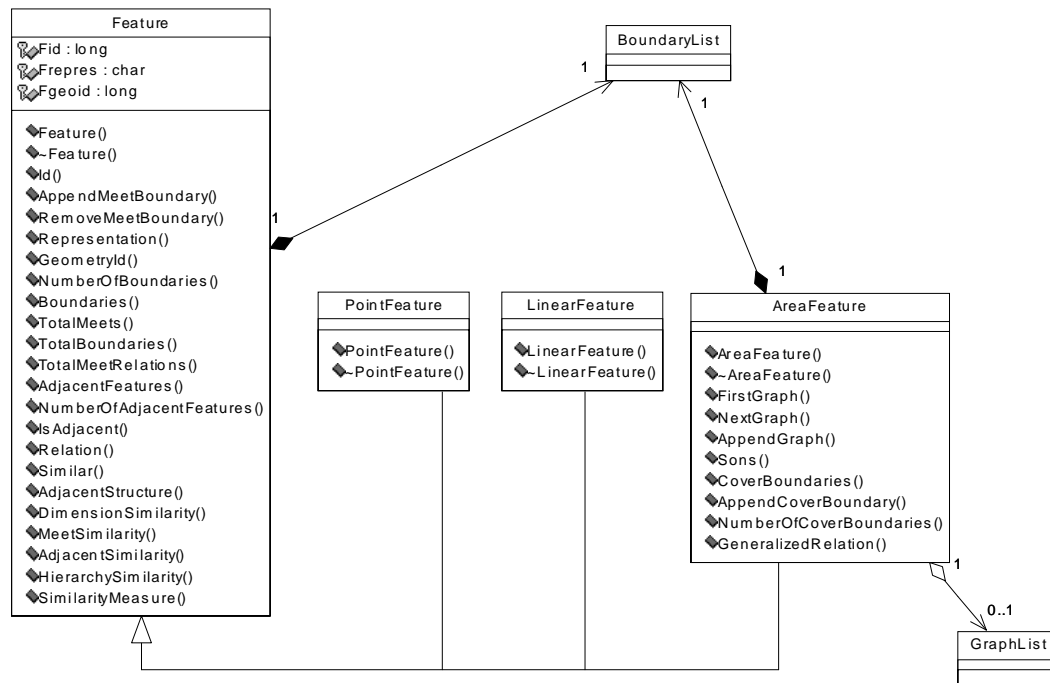


Figure 6.5: Diagram for the *Feature* class and its child classes.

6.2.2 Boundary

A boundary describes the intersection between two features. The attributes type and complement relationship are related to the component invariants of topological relations (Egenhofer and Franzosa 1993). The boundary type between connected features at the same level of a hierarchy will be some of the *meet* relations, while the boundary type between an area feature and its covered features will be some of the *cover* relations. [Figure 6.6](#) shows the specification for the *Boundary* class. Besides the pointer to the two adjacent features, this class has a list of identifiers that point to the geometric coordinates. The relation-based model uses just symbolic information, but this list of coordinates is provided for situations where the geometry is available for visualization purposes. The

topologic checker uses the symbolic information of the relation-based model to analyze the equivalent relations.

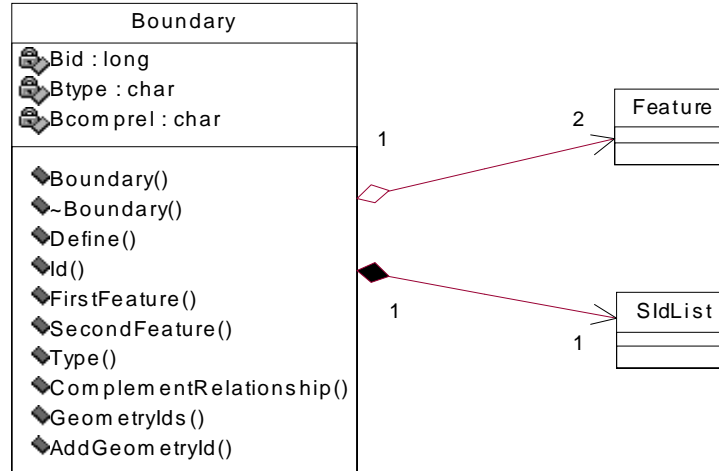


Figure 6.6: Diagram for *Boundary* class.

6.2.3 Graph

Each graph describes the connected set of features or an isolated feature. [Figure 6.7](#) shows the class diagram for a graph. It contains an attribute describing its identifier, a list of features, and a list of boundaries describing the adjacent intersections. Furthermore, it contains two pointers that permit us to navigate the hierarchical structure of the spatial scene. The first pointer points to the parent graph in the hierarchy, while the second pointer points to the area feature on the parent graph that contains this graph.

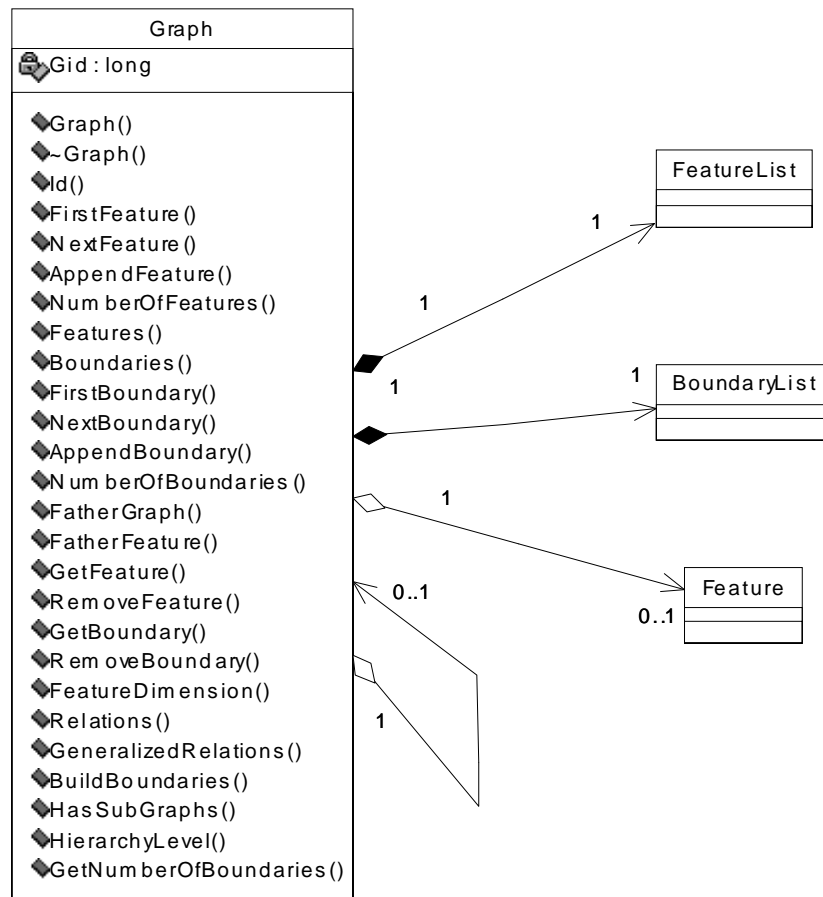


Figure 6.7: Diagram for *Graph* class.

6.2.4 Spatial Scene

A spatial scene contains a collection of spatial objects that may have multiple representations. [Figure 6.8](#) shows the diagram with the components of the *SpatialScene* class. It is basically composed of a list of graphs that describe each level of the hierarchy, and attributes that describes the next available identifiers for graph, feature, and boundary. There are some operations for comparing the topology and for comparing the detailed similarity between different spatial scenes, which are described in more detail in Appendix.

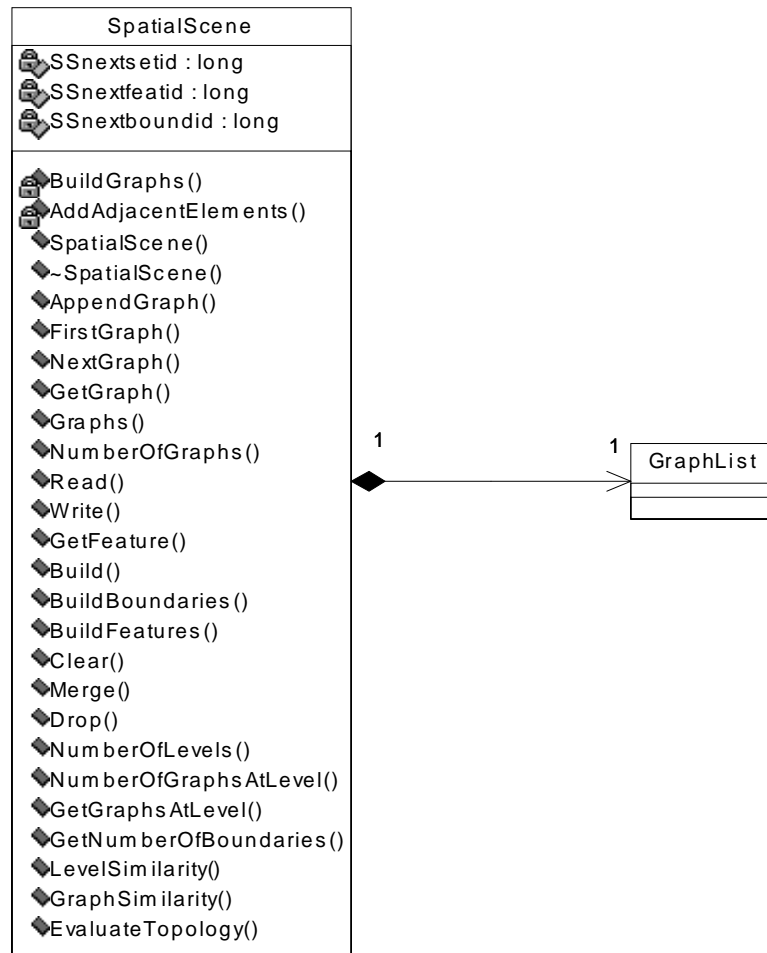


Figure 6.8: Diagram for *SpatialScene* class.

6.3 Additional Classes

The previous classes describe the relation-based model. The topological consistent checker uses some additional classes to represent isomorphic configurations and to represent the association graph. An isomorphic configuration is composed of nodes that contain pairs of features (one feature of each scene) that represent a match. [Figure 6.9a](#) shows a class representing a matching between two features, and [Figure 6.9b](#) shows the

isomorphic configuration class, as well the association graph class, which contains the possible initial matches based on the spatial relations between the spatial representations.

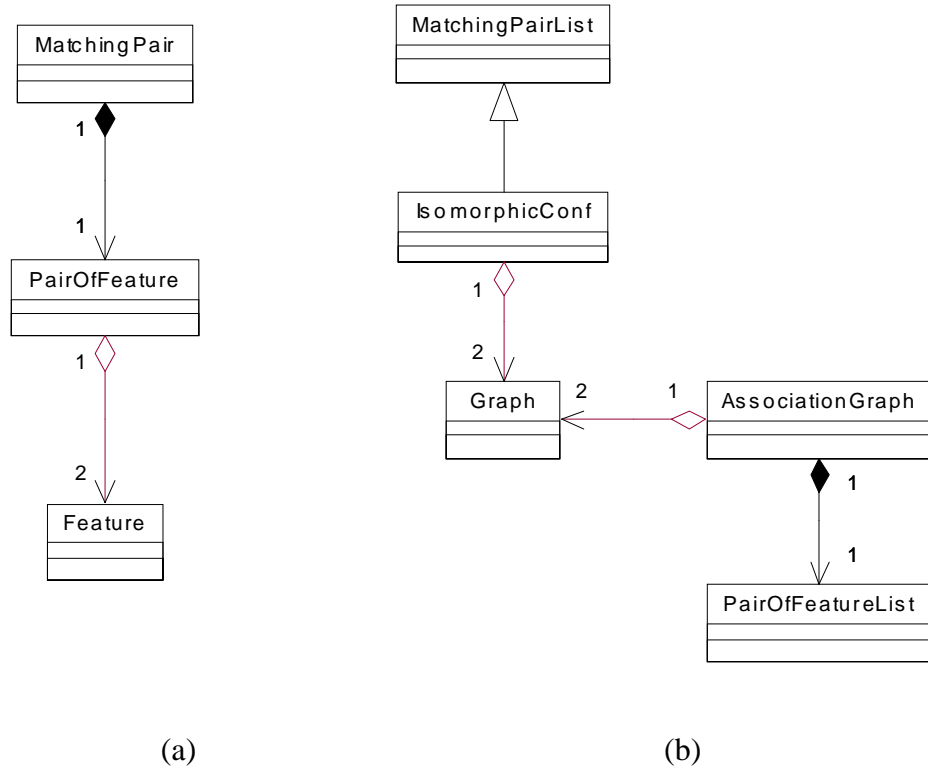


Figure 6.9: Class diagrams of additional classes used for the topological checker.

6.3.1 Matching Pair

The topological checker finds isomorphic configurations, but these configurations are not necessarily equivalent. The *MatchingPair* class (Figure 6.10) provides an attribute that identifies if the matching is valid in terms of the boundary sequence.

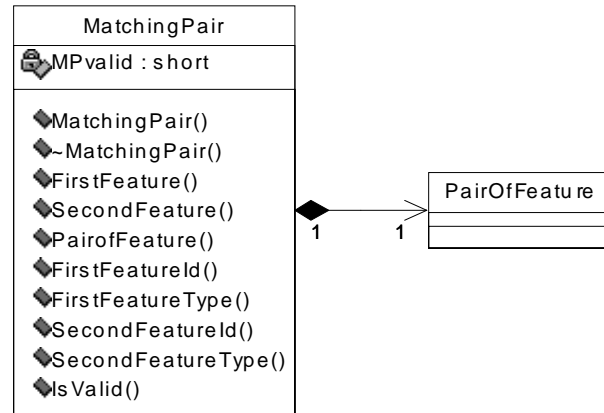


Figure 6.10: Diagram with attribute and operations for *MatchingPair* class.

6.3.2 Isomorphic Configuration

Figure 6.11 shows the operations for a class representing an isomorphic configuration. Each isomorphic configuration is related to two graphs (one for each spatial scene in the analysis). Operations to compare similarity between isomorphic configurations are available, as well as an operation to verify if the boundary sequences for the features in each node are compatible.

6.3.3 Association Graph

The association graph (Figure 6.12) describes the initial possible matches between representations of both scenes. It is built based on spatial constraints defined for the specific analysis. The association graph is composed of a list of feature pairs that are part of the graphs being compared. Once the scene graphs are defined, this class provides operations to build the possible matching pairs, as well to find the isomorphic configurations.

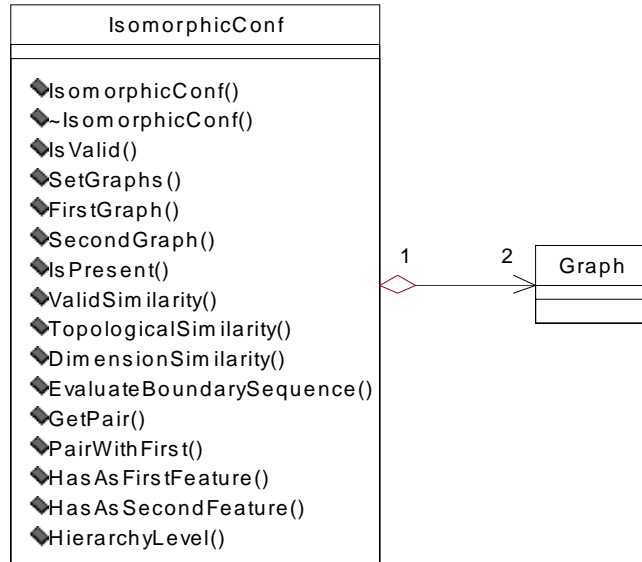


Figure 6.11: Diagram for *IsomorphicConf* class.

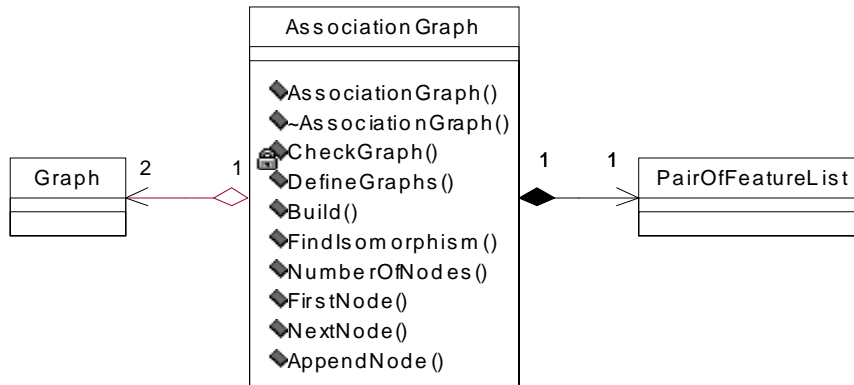


Figure 6.12: Diagram for the *AssociationGraph* class.

6.4 SPRING Model

SPRING (INPE/DPI 1997) is a GIS and Remote Sensing Image Processing system developed by the National Institute for Space Research (INPE 1997) in Brazil. It

integrates models for raster and vector data in one environment. It is implemented using the object-oriented paradigm. The vector model is based on the theory of cell complexes. This section is intended to show how to convert this GIS model into the relation-based model proposed in this thesis. The conceptual model of SPRING is divided into four abstraction levels ([Figure 6.13](#)):

- Real world level, which contains the real world categories of data to be modeled, such as soil maps, cadastral maps, geophysical and topographical data.
- Mathematical or conceptual level, which contains the formal definitions for entities of different models. It is based on the concept of geographic fields and geographic objects (Goodchild 1992). Classes for geographic features (fields and objects) and their specializations (digital terrain models and images, thematic and cadastral maps are defined at this level.
- Representation level, which maps the formal entities onto their geometric representations (one or multiple) that may vary through different scales and cartographic projections.
- Implementation level, which contains the data structures and algorithms to manipulate the data.

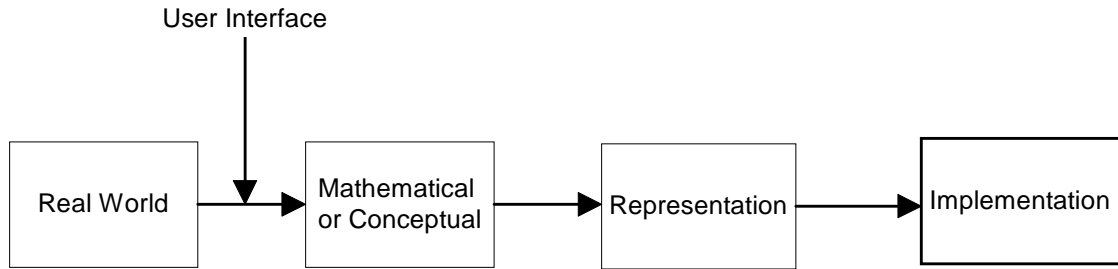


Figure 6.13: SPRING abstraction levels.

The conceptual class hierarchy of SPRING is shown in [Figure 6.14](#). The database contains a collection of workspaces representing projects of different areas and different projections. It may contain spatial and non-spatial objects with attributes. Each project may have several information layers describing the geographic data. Each layer represents a geographic field or a geographic collection of objects (object map). Specializations of geographic fields are thematic images, digital terrain models, and satellite images. Specialization of object maps are networks, and cadastral objects.

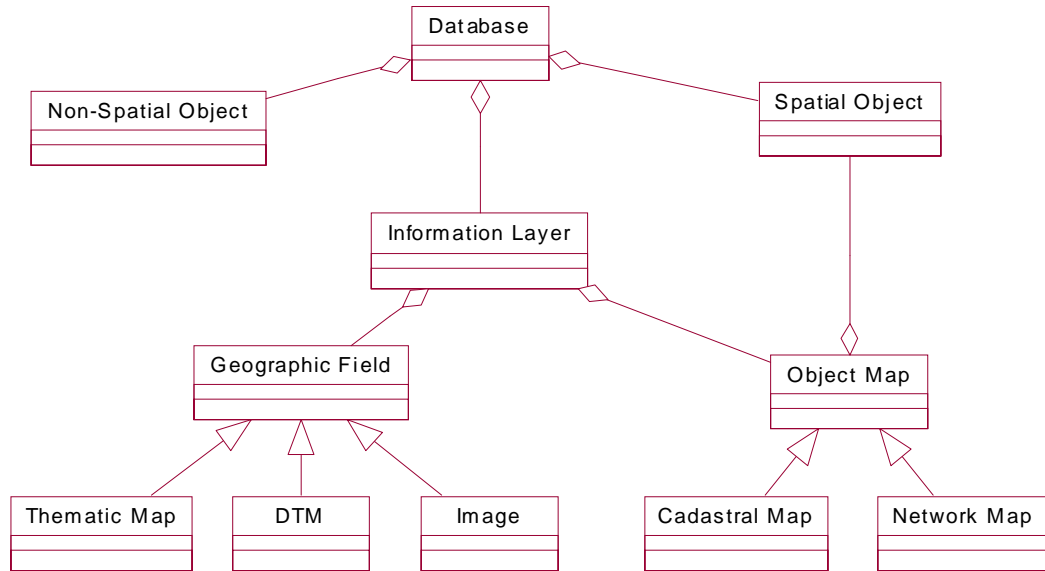


Figure 6.14: SPRING's conceptual model.

6.4.1 Converting SPRING's Vector Model into the Relation-Based Model

The relation-based model deals with the multiple representations of spatial objects. SPRING's model supports multiple representations in terms of data format and for spatial objects. Figure 6.15 shows the representation model of SPRING. Each information layer may have multiple representations in terms of data format. The raster representation is related to continuous space, and the vector representation describes contours of spatial objects. Within the vector representation, each spatial object may have several geometric representations, which can be translated into the symbolic representation of the relation-based model.

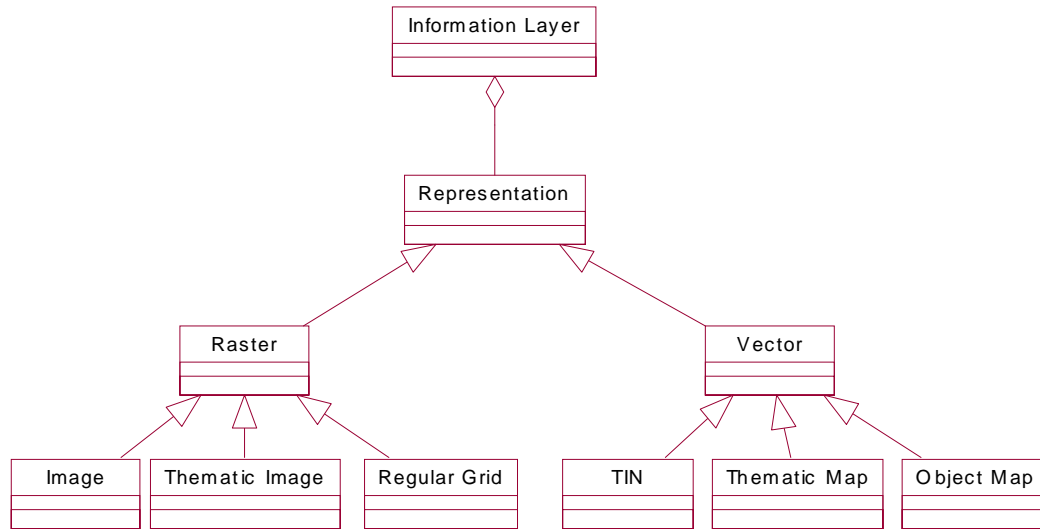


Figure 6.15: Representation model of SPRING.

The vector model of SPRING may describe the following data:

- Thematic Maps, containing regions that are geographically defined by one or more polygons associated to only one theme.
- Cadastral Maps, whose elements are regions, lines, and points for the representations of spatial objects.
- Network Maps, containing linear representations for spatial objects.
- Digital Terrain Models, as digital representations of a continuously distributed phenomenon referred as TIN (triangular network).

The SPRING vector model uses the cell complexes structure to describe the object representations, and the relationship between the representations are explicitly stored in the geometric cells. For example, a 2-cell (region or polygon) points to its 1-cells, and each 1-cell points to the two 2-cell to which it may belong. Each 1-cell points to the start and the end 0-cell that are the initial and final points of the 1-cell. The vector data

structure is organized into sets for each primitive geometry. These sets implement persistent elements. [Figure 6.16](#) shows the data structure for the vector model of SPRING. There are relationships between nodes and lines (0-cells and 1-cells), and between polygons and lines (2-cells and 1-cells). The polygons store the other polygons that they contain, which defines a hierarchical structure. The hierarchy exists between polygons. To include into the hierarchy lines that do not belong to polygons and isolated points, additional computation is necessary. The multiple representations of the objects are obtained through anchors that point to the primitive representations.

When converting this vector structure into the relation-based structure the basic procedures presented in Chapter 4 are used. A vector class of the SPRING model generates a spatial scene class of the graph model. The procedure to build graphs is implemented in a recursive way using the adjacent polygons of one polygon. Considering a vector model representing regions the following steps need to be addressed in the implementation of a converter:

- Each polygon of the polygon set is a graph feature of type area;
- Building the graphs: for each polygon P check if it is already assigned to one graph. If not, create a graph G , create an area feature AF representing this polygon, and insert it into graph G . Pick an adjacent polygon for this current polygon P , make it the current polygon P , create an equivalent area feature AF , and add it to current graph G . Build the complete graph G by recursively picking the adjacent polygons for the current polygon P . Update hierarchical links (graph parent and area feature parent) when necessary;
- Building the graph boundaries: for each feature on the graph, use [Algorithm 4.2](#) to convert a region into an area feature.

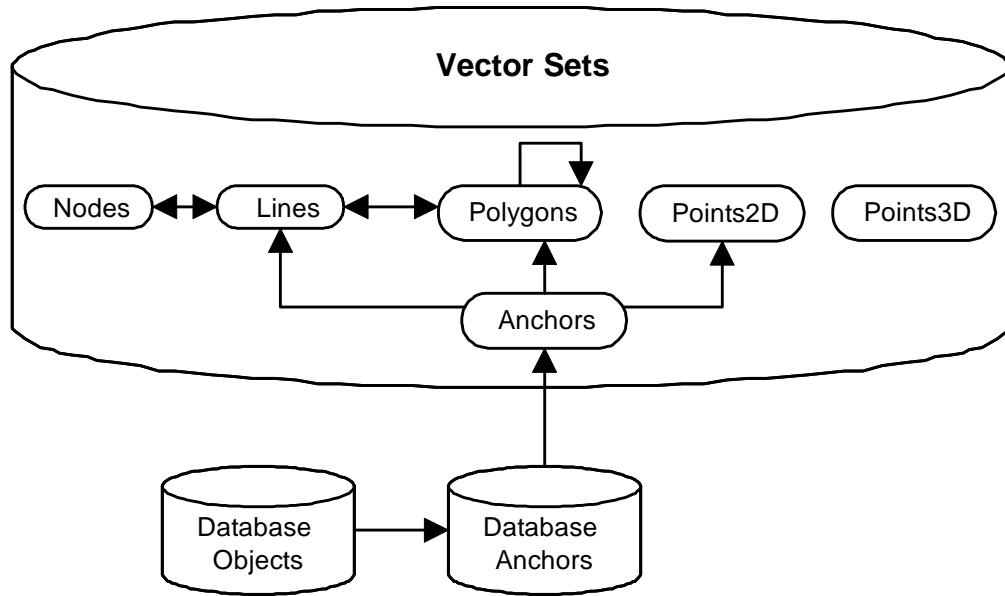


Figure 6.16: Vector data structure in SPRING.

6.4.2 Database Schema

The information in the relation-based model can be used as a metadata description for the spatial relations between spatial objects. The symbolic information incorporated into the model allow us to answer topological queries without access to geometric information. This symbolic model can be incorporated into the SPRING model as an additional representation type. Figure 6.17 shows the modified representation schema for SPRING with the addition of the *RelationBased* representation, which is equivalent to class *SpatialScene* of Figure 6.3. This type of representation is useful in situations where the geometry is missing. In cases where the geometry exists, it can be used as metadata description of spatial relations to speed up the processing of topological queries, as well as to perform high-level analyses of similarity or equivalence.

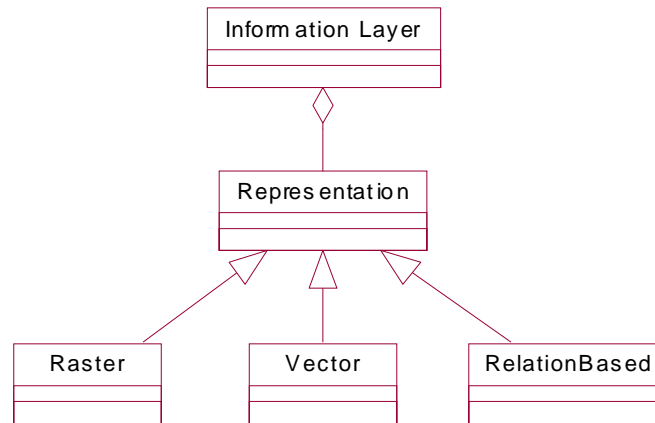


Figure 6.17: Modified representation model for SPRING.

The database schema to support this relation-based representation in SPRING is shown on [Figure 6.18](#) with links between relations. Each information layer may have several representations, and there is a link between the *InformationLayer* relation and the *Representation* relation through the identifier of the layer. The *RelationBased* representation is described by the relations *Graph*, *Feature*, and *Boundary*. The *Graph* relation contains the identifier of the graph, the representation identifier to which it belongs, and the feature identifier that contains or covers this graph (if it exists). The *Feature* relation attributes are the identifier of the feature, its type (area, line, or point), and the graph identifier. The *Boundary* relation describes the intersections between features. Its attributes are the boundary identifier, the first feature identifier, the second feature identifier, the sequence order of this boundary in the list of boundaries for the first feature, the sequence order of this boundary in the list of boundaries for the second feature, the boundary type, the complement relationship value, the geometry identifier, and the vector identifier that contains this geometry. The two last attributes may be provided as a way to link the symbolic information with the geometry. The two boundary

features are not necessarily in the same information layer, therefore, the boundary type may vary for meet, covers, and crosses relations. The multiple representations of the spatial objects can be obtained by linking the relation *SpatialObject* with the *Feature* relation through the identifier of the feature. This is a simpler and faster connection between objects and representations than the actual connection used with objects and anchors in the vector model of SPRING.

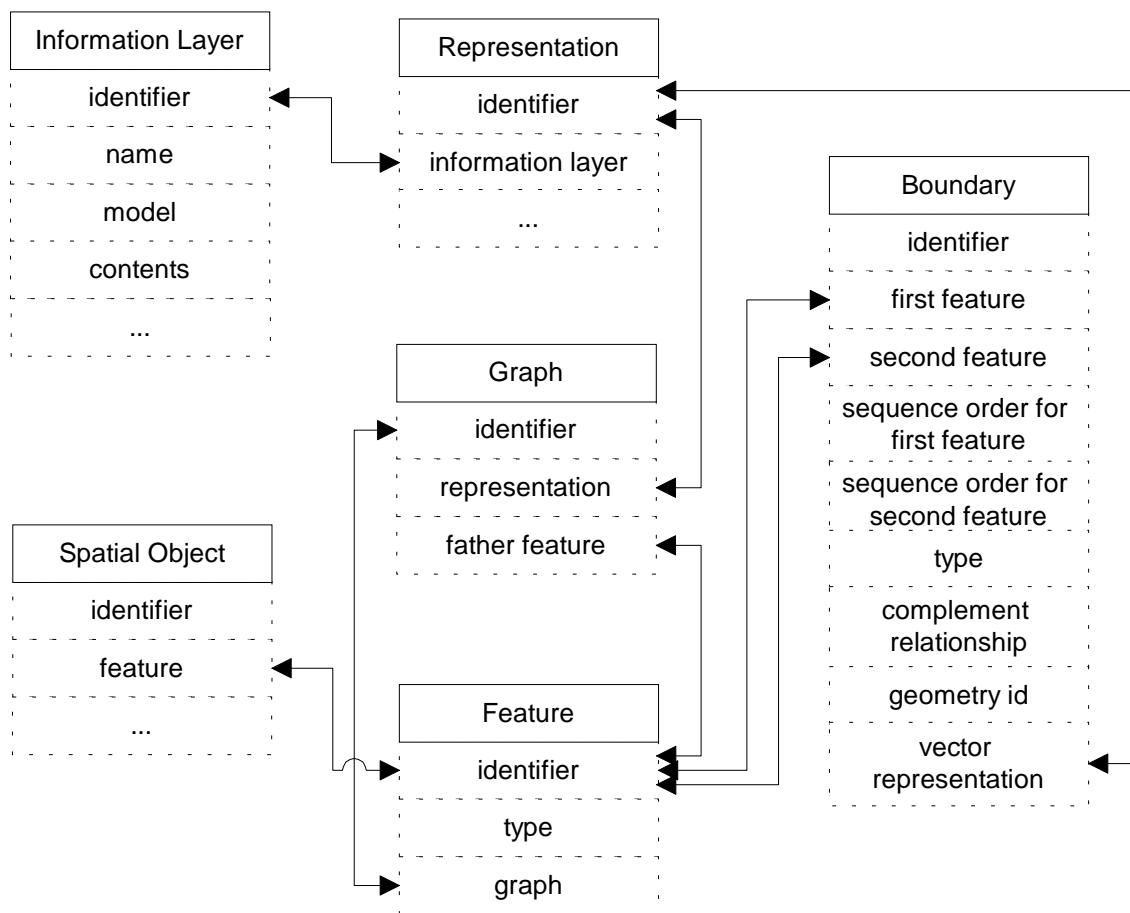


Figure 6.18: SPRING database schema to support the relation-based representation.

6.5 Summary

This chapter described the class structure for the relation-based model using object-oriented techniques. The UML notation leads to a clean design and specification. In addition to the basic classes that support the relation-based structure, this chapter showed how to integrate this representation into the existing GIS and Image Processing software SPRING, developed by the Image Processing Division of National Institute for Space Research (INPE) in Brazil. The addition of this type of representation into the system enables fast spatial inferences, allows for representations of incomplete information, and links closely to generating verbal instructions and verbal descriptions of spatial configurations.

Chapter 7

Conclusions

The main motivation of this work was to identify the model components to support the analysis of topological equivalence between spatial objects with multiple representations. There is a large amount of spatial data available for the same geographic areas, which originated from different sources, and current GISs lack methods to maintain consistency among multiple representations of geographic objects. Inconsistencies among multiple representations create contradictory information, which when passed to the decision level may result in wrong interpretations.

7.1 Summary

This thesis developed a qualitative model to represent spatial scenes composed of spatial objects with multiple representations. This qualitative model abstracts away the details of the geometric representations for spatial objects and focuses primarily on the spatial relations among the object representations or features. It translates a vector representation structure into a symbolic representation that captures the notion of the geometry based on the spatial relations modeled. A spatial scene is organized into a hierarchical structure of graphs, in which the nodes represent the object representations and the links describe the

topological relations between these representations. A topology checker has been developed based on this model, which supports the analysis of equivalence and similarity between spatial scenes. A set of similarity measurements between individual representations allows us to reason about changes that may affect spatial objects and consequently spatial scenes. The topology checker uses an association graph that contains initially possible matches between features of two scenes. These initial possible matches reflect the constraint model applied, which may vary from equivalence to some degree of similarity.

7.2 Major Findings

The major results related with the qualitative model developed in this thesis are:

- *Expressive power of relation-based model versus Cell complexes*

The relation-based model is a simplified representation of the cell complex structure. It stores the necessary topological information for checking topological similarity or to answer topological queries. The graph representation only stores the highest-dimensional object representations, while in a cell complex representation all cartographic elements –points, lines, areas– are explicitly stored. When converting a 2-cell into an area feature of the graph, only those 1-cells of this 2-cell that are part of another 2-cell are translated into boundary components in the graph structure. Considering a map representing polygonal areas, the total number of boundaries of type *1-meet* is equal to the number of 1-cells on the cell complex minus the number of 1-cells that are part of just one 2-cell. The space savings on the graph structure are more significant for a map with several isolated regions. In this

case these regions are translated as isolated features on the graph model without boundary components.

- *Complexity of updates*

The relation-based model saves space compared to the cell complex structure, however, in terms of operations complexity for merging or splitting elements they are similar. The topological structure of cell complexes is affected by changes on 1-cells, which is equivalent to changing 1-dimensional boundaries in the graph model. There is a direct relationship between 1-cells that are part of two 2-cells and 1-dimensional boundaries of area features in the graph. A merge operation in the cell complex structure corresponds to deleting the 1-cell(s) between the two 2-cells. This operation in the graph model corresponds to deleting the common boundaries between the two area features. When merging two regions in a cell complex, it is necessary to replace them by a single one, and to update the border of the resulting region as well as the border of its adjacent regions. In a equivalent way, the merge operation in the graph corresponds to substituting the two area nodes by one area node and updating the boundary sequence for the new node as well as for its adjacent nodes. Therefore, there is an equivalence between the sequence of 1-cells that describe a 2-cell, and the sequence of boundary components that describes an area feature in the graph. Similarly to the merge operation, the updated topology after a split operation can be derived using the 1-cells for the cell complex, and using the boundaries for the graph structure.

- *Use as metadata description for spatial relations*

The relation-based information is an effective way for representing spatial objects, and not just as a temporary representation to help on the process for comparing topology between spatial scenes. It can be easily incorporated into an existing GIS, as explained for SPRING (INPE/DPI 1997) case, and it can be used as metadata descriptions of spatial relations in digital libraries, avoiding the expensive computation of spatial relations using the geometric information.

- *Reasoning capabilities*

The relation-based model enables fast spatial inferences, allows for representations of incomplete information, and links closely to generating verbal instructions and verbal descriptions of spatial configurations. Cell complexes are more a computational model composed of building blocks to represent the topology from a cartographic point of view, while the relation-based model comes closer to non-geographical languages.

- *Computational complexity*

The topology checker developed based on the relation-based model looks for isomorphic configurations in spatial scenes. Isomorphism is an NP-complete problem, i.e., there is no optimal solution in a reasonable amount of time when considering all mappings for n nodes in both scenes. The computation is proportional to $n!$ if all possible node mappings are tried. However, by modeling the graph with the spatial relationships between features and

applying constraints of equivalence and similarity to build the association graph, it is possible to find matches in a reasonable amount of time, as the number of possible matches is restricted to a small amount. This computation time is related to the number of nodes in the association graph and how these nodes are linked. This topological checker is an important tool, because it provides an automatic analysis of geographic databases containing multiple representations for spatial objects, that may be represented at different scales, giving support for the test of new generalization algorithms.

7.3 Future Work

Progress has been made in defining and developing qualitative models to represent geographic phenomena. However, there are several areas that need more intensive investigation or development. Future work should concentrate on the integration of topological models with others types of spatial relations such as metric, directional, and semantic. Use of the relation-based model for linear representations such as homogeneous and heterogeneous networks should be investigated. Another important topic for future investigation is how to formalize and translate human knowledge about cartographic generalization into information and constraints for the qualitative models. Finally, it is important to put attention on the aspects of multi-modal spatial querying (Egenhofer 1996), and identify how to translate information from sketches and verbal descriptions into spatial qualitative models, as well as how to generate a sketch from the qualitative information. These interesting research questions are addressed below.

7.3.1 Integration of Spatial Relationships

The relation-based model expresses topological relationships between spatial object representations. In order to characterize the essence of spatial relations, it is necessary to develop a global model that integrates information about topology, direction, metric, and semantic aspects. The metric information may be represented in a quantitative way. Qualitative metric relations, for example, can be specified based on a range of discrete distance values (Hong 1994). Topologic information is important during searches in a spatial database, however, parameters related to distances, directions, and semantic values can be used as refinements for the spatial search. For example, topological information permits us to identify containment relations between object representations, but we do not know how much of the contained object representations are inside the object representation that contains them. Shariff (1996) introduced a set of metric refinements, which describe relationships between boundaries, that can be used to identify measures between the inner and outer object representations. For situations between disjoint object representations, the topological information captures neither the relative position between them nor how far apart they are. Qualitative information about directions between objects, known as cardinal directions (Frank 1991), can be used to refine the position relationship, while the metric refinements can be used again to identify more precisely the distance relationships. In terms of object attributes, a semantic network describing relationships between feature types can be built, with weights at nodes that will indicate the degree of difference between linked features.

The integration of these four elements –topology, direction, metric, and semantics– should take into consideration the idea of conceptual neighborhoods in order to define levels of similarity between spatial scenes. Similarity analysis gradually replaces spatial

relations in a scene by their conceptual neighbors, in an attempt to construct one scene from another. Conceptual neighbors describe the gradual changes for spatial relationships. They have been studied for each of these individual elements separately. It is of interest to analyze if the combination of these four elements generates new sets of conceptual neighbors. Additional testing needs to be done to verify if such gradual changes match with human intuition, in the tradition of earlier evaluations and calibrations of natural-language spatial relations (Mark et al. 1995). These conceptual neighbors can be used as relaxation rules during the process of creating nodes for the association graph containing possible feature matches for different spatial scenes.

7.3.2 Extension to Linear Features

The relation-based model has been described using polygonal data, however, it can be used in a similar way to describe homogeneous or heterogeneous networks. A homogeneous network is characterized by connected linear elements, and the equivalent cell complex contains 1-cells representing these linear elements, and 0-cells (extreme points of 1-cells) that correspond to junctions between the linear elements or even an end or start point of some linear element. When converting this cell complex structure into the relation-based model, the 1-cells become linear features on the graph model, and the 0-cell intersections become the boundary components in the graph model. For networks that contain linear and polygonal elements (heterogeneous network), the 1-cells that are not part of an area element should be converted into the graph model in the same way as the homogenous network. The area elements should be converted into area features in the graph model with the boundary components being described by the intersections of its 1-

cells with other 1-cells of the network. The topological invariants for lines (Clementine and Di Felice 1998) can be used to model the arc properties of the graphs.

7.3.3 Integration with Model Generalization

Map generalization is usually associated with simplification of line shapes. However, in several maps many of these lines together represent different features that are associated through spatial relations. It is important that line generalization algorithms try to take into consideration the general structure of the data, and not over simplify each individual line at a time without considering if it is a part of more complex feature. This generalization approach is essentially metric, as it concentrates on simplification of line shapes, and consists of several transformations at the geometrical level that are traditionally performed by cartographers. Another type of generalization, called model generalization, is based on topological aspects concerning the map structure and topological relationships between entities. The model generalization approach should include spatial relations and semantic information, and give support to the development of rules to check inconsistencies of data represented at different scales.

The cartographic generalization is a quantitative approach based on metric information extracted from line shapes, while model generalization is a qualitative approach more concerned with the general structure of the data. There has been some work related to model generalization (Buttenfield 1995; Ruas and Lagrange 1995), and this is an area of continuing research. It is important to clearly identify conceptual operators and rules associated with them that are shape-independent. Since this thesis work developed a qualitative model to describe spatial objects with multiple representations, it would be interesting to use the development of conceptual models of

generalization as information and constraints to describe qualitative models for representing spatial data.

7.3.4 Interaction with Multi-Modal Languages

Users often have different conceptual views for the same geographic objects, which makes it difficult to model spatial queries. Current spatial queries are mostly based on non-spatial alphanumeric command languages such as SQL, which requires lots of time during training. Extended versions of SQL have been described to perform spatial queries (Ingram and Philips 1987; Herring et al. 1988; Egenhofer 1994). Egenhofer (1992) identified some drawbacks about using SQL for spatial queries, and the use of graphics described by sketches and speech, promises to be a more intuitive and precise way to specify spatial queries (Egenhofer 1996).

Sketch-and-talk is a new way to query spatial databases, and it uses graphic and voice supporting more directly human spatial thinking. Natural language descriptions (Talmy 1983) captures the topological properties of a spatial scene, but abstracts away details about directions and distances. Sketch-and-talk is an alternative form to specify spatial queries, as it incorporates additional constraints related with direction and metric properties. An interesting research topic to address is how to efficiently translate sketch and talk descriptions into the relation-based model, as well how to generate a sketch from the relation-based information for visual purposes. There may be situations in which only the qualitative information is available, and a graphical representation is needed. The resultant sketch from the qualitative information will not be metrically accurate, but would capture the topological aspects of the scene. Considering a polygonal data, each node feature of the graph will correspond to a region in the sketch, and each component

of the boundary sequence of a feature will define a *meet* relation between two regions in the sketch. The sketching process gets more complicate if the number of adjacent components between two regions is more than one. Finally, the levels of hierarchy in the graph will define containment relations in the sketch, and a higher-order area feature in the graph will *contains* or *covers* all of its lower-level features in the sketch.

References

- R. Abler (1987). The National Science Foundation National Center for Geographic Information and Analysis. *International Journal of Geographical Information Systems* 1(4): 303-326.
- A. V. Aho, J. E. Hopcroft and J. E. Ullman (1974). *The Design and Analysis of Computer Algorithms*. Addison Wesley, Reading, MA.
- P. Alexandroff (1961). *Elementary Concepts of Topology*. Dover Publications Inc., New York, NY.
- M. P. Armstrong (1991). Knowledge Classification and Organization. in: B. Buttenfield and R. McMaster (Eds.), *Map Generalization: Making Rules for Knowledge Representation*, pp. 86-102, Longman, London.
- V. K. Balakrishnan (1997). *Graph Theory*. McGraw Hill.
- K. Beard and T. Smith (1997). A Framework for Meta-Information in Digital Libraries. in: A. Sheth and W. Klaus (Eds.), *Managing Multimedia Data: Using Metadata to Integrate and Apply Digital Data*, McGraw Hill.
- K. Beard (1988). How to Survive on a Single Detailed Database. in: *AutoCarto 8*, Baltimore, MD, pp. 211-220.

K. Beard (1989). Design Criteria for Automated Generalization. in: *International Cartographic Association Conference*, Budapest, Hungary, pp. 32-40.

K. Beard (1991). Constraints on Rule Formation. in: B. Buttenfield and R. McMaster (Ed.), *Map Generalization: Making Rules for Knowledge Representation*, pp. 121-135, Wiley, New York, NY.

K. Beard and W. A. Mackaness (1991). Generalization Operations and Supporting Structures. in: *Auto Carto 10*, Baltimore, MD, pp. 29-45.

B. Becker, H.-W. Six and P. Widmayer (1991). Spatial Priority Search: An Access Technique for Scaleless Maps. in: *SIGMOD 91*, pp. 128-137.

G. Booch (1994). *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, Inc., Reading, MA.

G. Booch, I. Jacobson and J. Rumbaugh (1996). The Unified Modeling Language for Object-Oriented Development, Document Set Version 0.91 Addendum UML Update, Rational Software Corporation, <http://www.rational.com>. [Accessed in September 1998].

K. Brassel and R. Weibel (1988). A Review and Conceptual Framework of Automated Map Generalization. *International Journal of Geographic Information Systems* 2(3): 229-244.

M. Brodie (1984). On the Development of Data Models. in: M. Brodie, J. Mylopoulos and J. Schmidt (Eds.), *On Conceptual Modelling: Perspectives, from Artificial Intelligence, Databases and Programming Languages*, pp. 19-48, Springer Verlag, New York, NY.

B. P. Bruegger and A. U. Frank (1989). Hierarchies over Topological Data Structures. in: *ASPRS/ACSM Annual Convention*, Baltimore, MD, pp. 137-145.

B. P. Bruegger and W. Kuhn (1991). Multiple Topological Representations, Technical Paper 91-17, NCGIA, Department of Surveying Engineering University of Maine.

T. Bruns and M. Egenhofer (1996). Similarity of Spatial Scenes. in: M. J. Kraak and M. Molenaar (Eds.), *7th International Symposium on Spatial Data Handling*, Delft, The Netherlands, Taylor & Francis, pp. 4A.31-4A.42.

G. L. Bundy, C. B. Jones and E. Furse (1995). Holistic Generalization of Large Scale Cartographic Data. in: J.-C. Muller, J.-P. Lagrange and R. Weibel (Eds.), *GIS and Generalization Methodology and Practice*, pp. 106-119, Taylor & Francis, London.

P. A. Burrough (1986). *Principles of Geographical Information Systems for Resources Assesment*. Clarendon Press, Oxford.

B. Buttenfield (1989). Multiple Representations: Initiative 3 Specialist Meeting Report, National Center for Geographic Information and Analysis, Technical Report 89-3, NCGIA, UCSB Santa Barbara, CA.

B. P. Buttenfield (1989). Scale-Dependence and Self-Similarity in Cartographic Lines. *Cartographica* 26(1): 79-100.

B. P. Buttenfield (1991). A Rule for Describing Line Feature Geometry. in: B. Buttenfield and R. McMaster (Eds.), *Map Generalization: Making Rules for Knowledge Representation*, pp. 150-171, Wiley, New York, NY.

B. P. Buttenfield (1993). Multiple Representations – Closing Report, National Center for Geographic Information and Analysis - NCGIA, UCSB Santa Barbara, CA, Technical Report.

B. Buttenfield (1995). Object-Oriented Map Generalization: Modeling and Cartographic Considerations. in: J.-C. Muller, J.-P. Lagrange and R. Weibel (Eds.), *GIS and Generalization - Methodology and Practice*, pp. 91-105, Taylor & Francis, Bristol, PA.

G. Camara, U. Freitas, R. Souza, M. Casanova, A. Hemerly and C. B. Medeiros (1994). A Model to Cultivate Objects and Manipulate Fields. in: *Proceedings 2nd ACM Workshop on Advances in GIS*, pp. 20-28.

V. Chacra, P. M. Ghare and J. M. Moore (1979). *Applications of Graph Theory Algorithms*. Oxford, North Holland, New York.

E. Clementine and P. Di Felice (1998). Topological Invariants for Lines. *IEEE Transactions on Knowledge and Data Engineering* 10(1): 38-54.

J. P. Corbett (1979). Topological Principles of Cartography, Bureau of the Census, Department of Commerce, Technical Report, Washington , DC.

R. G. Cromley (1991). Hierarchical Methods to Line Simplification. *Cartography and Geographic Information Systems* 18(2): 125-131.

M. de Berg, M. van Kreveld and S. Schirra (1995). A New Approach to Subdivision Simplification. in: *ACSM/ASPRS Annual Convention, Autocarto 12*, Charlotte, NC, pp. 79-88.

J. Dettori and E. Puppo (1996). How Generalization Interacts with the Topological and Metric Structure of Maps. in: M. J. Kraak and M. Molenaar (Eds.), *7th International Symposium on Spatial Data Handling*, Delft, The Netherlands, Taylor & Francis, pp. 9A.27-9A.38.

T. Devogele, J. Trevisan and L. Raynal (1996). Building a Multi-Scale Database with Scale-Transition Relationships. in: M. J. Kraak and M. Molenaar (Eds.), *7th International Symposium on Spatial Data Handling*, Delft, The Netherlands, Taylor & Francis, pp. 6.19-6.33.

D. Douglas and T. Peucker (1973). Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Canadian Cartographer* 10(2): 112-122.

- M. Egenhofer (1994). Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering* 1(4): 86-95.
- M. Egenhofer (1996). Multi-Modal Spatial Querying. in: M. J. Kraak and M. Molenaar (Eds.), *7th International Symposium on Spatial Data Handling*, Delft, The Netherlands, Taylor & Francis, pp. 785-799.
- M. Egenhofer and K. Al-Taha (1992). Reasoning about Gradual Changes of Topological Relationships. in: A. U. Frank, I. Campari, and U. Formentini (Eds.), *Theories and Methods of Spatial Temporal Reasoning*, Pisa, Italy, pp. 196-219.
- M. Egenhofer, E. Clementine and P. di Felice (1994). Evaluating Inconsistencies Among Multiple Representations. in: T. C. Waugh and R. G. Healey (Eds.), *Sixth International Symposium on Spatial Data Handling*, Edinburgh, Scotland, pp. 901-920.
- M. Egenhofer and A. Frank (1989). Object-Oriented Modeling in GIS: Inheritance and Propagation. in: *Autocarto 9*, Baltimore, MA, pp. 588-598.
- M. Egenhofer, A. Frank and J. Jackson (1989). A Topological Data Model for Spatial Databases. in: *Symposium on the Design and Implementation of Large Spatial Databases*, Santa Barbara, CA, pp. 271-286.
- M. Egenhofer and D. Mark (1995). Naive Geography. in: *Spatial Information Theory – A Theoretical Basis for GIS, International Conference, COSIT '95*, Semmering, Austria, Springer-Verlag, Berlin, pp. 1-15.

M. J. Egenhofer (1992). Why not SQL! *International Journal of Geographical Information Systems* 6(2): 71-85.

M. Egenhofer, E. Clementini and P. Di Felice (1994). Topological Relations between regions with holes. *International Journal of Geographical Information Systems* 8(2): 129-142.

M. Egenhofer and R. Franzosa (1991). Point-Set Topological Spatial Relations. *International Journal Geographical Information Systems* 5(2): 161-174.

M. Egenhofer and R. D. Franzosa (1995). On the Equivalence of Topological Relations. *International Journal of Geographical Information Systems* 9(2): 133-152.

M. Egenhofer and J. Herring (1990). A Mathematical Framework for the Definition of Topological Relationships. in: K. Brassel and H. Kishimoto (Eds.), *Fourth International Symposium on Spatial Data Handling*, Zurich, Switzerland, pp. 803-813.

M. Egenhofer and J. Herring (1991). Categorizing Binary Topological Relationships Between Regions, Lines and Points in Geographic Databases, Department of Surveying Engineering, University of Maine, Orono, ME.

A. Frank and W. Kuhn (1986). Cell Graph: A Provable Correct Method for the Storage of Geometry. in: *Second International Symposium on Spatial Data Handling*, Seattle, WA, pp. 411-436.

A. Frank (1991). Qualitative Spatial Reasoning about Cardinal Directions. in: *Autocarto 10*, Baltimore, MD, pp. 148-167.

C. Freska (1992). Using Orientation Information for Qualitative Spatial Reasoning. in: *Theories and Methods of Spatial-Temporal Reasoning in Geographic Space*, Springer-Verlag, pp. 162-178.

M. Goodchild (1992). Geographical Information Science. *International Journal of Geographical Information and Analysis* 6(1): 31-45.

J. L. Gross and T. W. Tucker (1987). *Topological Graph Theory*. Wiley Interscience, New York, NY.

S. C. Guptill (1990). An Enhanced Digital Line Graph Design: A Feature-Based Model for Digital Spatial Data Bases that Represent Geographic Phenomena. Washington, DC, US Geological Survey.

R. M. Haralick and L. Shapiro (1979). The Consistent Labeling Problem: Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1(2): 173-184.

R. M. Haralick and L. Shapiro (1980). The Consistent Labeling Problem: Part II. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2(3): 193-203.

R. M. Haralick, L. Shapiro and J. Campbell (1985). Extraction of Drainage Networks by Using the Consistent Labeling Technique. *Remote Sensing of Environmet* 18(2): 163-175.

J. Herring (1987). TIGRIS: Topologically integrated geographic information systems. in: N.R. Chrisman (Ed.), *Autocarto 8*, Bethesda, MD, pp. 282-291.

J. Herring, R. Larsen and J. Shivakumar (1988). Extensions to the SQL Language to Support Spatial Analysis in a Topological Data Base. in: *GIS/LIS*, San Antonio, TX, pp. 741-750.

J.-H. Hong (1994). *Qualitative Distance and Direction Reasoning in Geographic Space*, PhD. Thesis, University of Maine, Department of Surveying Engineering.

K. Ingram and W. Philips (1987). Geographic Information Processing Using a SQL-Based Query Language. in: N.R. Chrisman (Ed.), *Auto-Carto 8, Eight International Symposium on Computer Assisted Cartography*, Baltimore, MD, pp. 326-335.

INPE (1997). Instituto Nacional de Pesquisas Espaciais. <http://www.inpe.br>.

INPE/DPI (1997). SPRING – GIS and Remote Sensing Image Processing System. <http://sputnik.dpi.inpe.br/spring>.

I. Jacobson (1992). *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, Reading, MA.

J. P. Jackson (1989). Algorithms for Triangular Irregular Networks Based on Simplicial Complex Theory. in: *ACSM/ASPRS Annual Convention*, Baltimore - MD, pp. 131-136.

G. A. James, A. D. Cliff and P. Haggett (1970). Some Discrete Distributions for Graphs with Applications to Regional Transport Networks. *Geografiska Annaler* 52B: 14-21.

C. B. Jones, L. B. Geraint and J. M. Ware (1995). Map Generalization with a Triangulated Data Structure. *Cartography and Geography Information Systems* 22(4): 317-331.

C. B. Jones, D. B. Kidner, Q. Luo, G. L. Bundy and J. M. Ware (1996). Database Design for a Multi-Scale Spatial Information System. *International Journal Geographical Information Systems* 10(8): 901-920.

D. B. Kidner and C. B. Jones (1994). A Deductive Object-Oriented GIS for Handling Multiple Representations. in: T. C. Waugh and R. G. Healey (Eds.), *6th international Symposium on Spatial Data Handling*, Edinburgh, Scotland, UK, pp. 882-900.

D. E. Knuth (1968). *The Art of Computer Programming*. Addison-Wesley, Reading-MA.

B. Kuijpers, J. Paredaens and J. V. den Bussche (1995). Lossless Representation of Topological Spatial Data. in: Max J. Egenhofer and John Herring (Eds.), *Advances in Spatial Databases, 4th International Symposium SSD'95*, Portland - ME, Springer, pp. 1-13.

R. Laurini and F. Milleret-Raffort (1992). A Conceptual Framework for Multiple Representations of Spatial Objects with the Entity-Relationship Approach. *Computers, Environment and Urban Systems* 16: 299-311.

Z. Li and S. Openshaw (1992). Algorithms for Automated Line Generalization Based on a Natural Principle of Objective Generalization. *International Journal of Geographical Information Systems* 6(5): 373-389.

W. Mackaness and K. Beard (1993). Use of Graph Theory to Support Map Generalization. *Cartography and Geographic Information Systems* 20(4): 210-221.

D. Mark, D. Comas, M. Egenhofer, S. Freundschuh, M. Gould and J. Nunes (1995). Evaluating and Refining Computational Models of Spatial Relations Through Cross-Linguistic Human-Subject Testing. in: A. U. Frank and W. Kuhn (Eds.), *Spatial Information Theory - A Theoretical Basis for GIS, International Conference COSIT 95*, Semmering, Austria, pp. 553-568. Lecture Notes in Computer Science, Vol. 998.

D. Mark (1989). Conceptual Basis for Geographic Line Generalization. in: *AutoCarto 9*, Baltimore, MD, pp. 68-77.

D. Mark (1991). Object Modelling and Phenomenon-Based Generalization. in: B. Buttenfield and R. McMaster (Eds.), *Map Generalization: Making Rules for Knowledge Representation*, pp. 103-118, Wiley, New York, NY.

C. R. F. Maunder (1996). *Algebraic Topology*. Dover.

R. McMaster and S. Shea (1992). *Generalization in Digital Cartography*. American Association of Geographers, Washington.

R. McMaster (1989). The Integration of Simplification and Smoothing Algorithms in Line Generalization. *Cartographica* 26(1): 101-121.

R. B. McMaster (1991). Conceptual Frameworks for Geographical Knowledge. in: B. Buttenfield and R. McMaster (Eds.), *Map Generalization: Making Rules for Knowledge Representation*, pp. 21-39, Wiley, New York, NY.

R. B. McMaster and K. S. Shea (1988). Cartographic Generalization in a Digital Environment: A Framework for Implementation in a Geographic Information System. in: *GIS/LIS*, San Antonio, TX, pp. 240-249.

R. McMaster and H. Veregin (1996). Multiple Representations of Spatial Data. <http://www.geog.umn.edu/umucgis/nomination4.html>. [Accessed in December 1997].

J.-C. Muller (1990). The Removal of Spatial Conflicts in Line Generalization. *Cartography and Geographic Information Systems* 17(2): 141-149.

J.-C. Muller (1991). Generalization of Spatial Databases. in: D. J. Maguirre, M. Goodchild and D. Rhind (Ed.), *Geographic Information Systems*, 1, pp. 457-475, Longman, London.

J. C. Muller and W. Zeshen (1992). Area-Patch Generalization: a Competitive Approach. *The Cartographic Journal* 29(2): 137-144.

J.-C. Muller, L.-P. Lagrange and R. Weibel (1995). *GIS and Generalization: Methodology and Practice*. Taylor & Francis, London.

J. Munkres (1975). *Topology, A First Course*. Prentice Hall, Englewood Cliffs, NJ.

NCDCDS (1988). The Proposed Standard for Digital Cartographic Data. *The American Cartographer* 15(1): 23-31.

NSDS (1993). Toward a Coordinated Spatial Data Infrastructure for the Nation. Washington, DC, Mapping Science Committee.

OGC (1998). OGC – Open GIS Consortium, Technical Specifications. <http://www.opengis.org/techno/specs.htm> [Accessed in March 1998], .

P. Oosterom (1989). A Reactive Data Structure for Geographic Information Systems. in: *Autocarto 9*, Baltimore, MD, pp. 665-674.

J. Paiva and M. Egenhofer (1995). Topological Equivalence of Regions with Holes: The Concepts and an Incremental Algorithm, Technical Report, National Center for Geographic Information and Analysis, NCGIA, University of Maine.

J. Paiva, M. Egenhofer and A. Frank (1992). Spatial Reasoning about Flow Directions: Towards an Ontology for River Networks. in: *Proceedings of the 17th International Society for Photogrammetry and Remote Sensing Congress Meeting, XXIX, Part B3 Commission III*, Washington, D.C., pp. 318-324.

D. Papadias and M. Egenhofer (1997). Algorithms for Hierarchical Spatial Reasoning. *GeoInformatica* 1(3): 251-273.

E. Puppo and G. Dettori (1995). Towards a Formal Model for Multiresolution Spatial Maps. in: Max J. Egenhofer and John Herring (Eds.), *Advances in Spatial Databases, 4th International Symposium, SSD95*, Portland, ME, Springer, pp. 152-169. Lecture Notes in Computer Science 951.

H. S. Ranganath and L. J. Chipman (1991). *A Graph Theoretic Approach to Scene Matching*, Technical Report, University of Alabama.

P. Rigaux and M. Scholl (1994). Multiple Representation Modeling and Querying. in: *IGIS94*. in: J. Nievergelt, T. Roos, H. Scheck and P. Widmayer (Eds.), *International Workshop on Advanced Research in Geographic Information Systems*, pp. 59-69. Lecture Notes in Computer Science, Vol. 884, Springer.

A. Ruas (1995). Multiple Paradigms for Automating Map Generalization: Geometry, Topology, Hierarchical Partitioning and Local Triangulation. in: *ACSM/ASPRS, Auto Carto 12*, Charlotte, NC, pp. 69-78.

A. Ruas and J. P. Lagrange (1995). Data and Knowledge Modeling for Generalization. in: J.-C. Muller, J.-P. Lagrange and R. Weibel (Eds.), *GIS and Generalization: Methodology and Practice*, pp. 73-90, Taylor & Francis, Bristol, PA.

J. Rumbaugh (1991). *Object-Oriented Modeling and Design*. Prentice Hall.

SAIF (1995). Spatial Archive and Interchange Format, Release 3.2.
<http://www.env.gov.bc.ca/gdbc/saif32/toc.htm>. [Accessed in October 1998].

SDTS (1992). Spatial Data Transfer Standard, Federal Information Process Standards Publication 173. Gaithersburg - MD, U.S. Department of Commerce.

R. Shariff (1996). *Natural Language Spatial Relations: Metric Refinements of Topological Properties*, PhD. Thesis, University of Maine, Orono, ME.

K. S. Shea and R. B. McMaster (1991). Cartographic Generalization in a Digital Environment: When and How to Generalize. in: *Auto Carto-9*, Baltimore, Maryland, pp. 56-67.

A. N. Strahler (1960). *Physical Geography*. New York: John Wiley and Sons.

L. Talmy (1983). How Language Structures Space. in: H. Pick and L. Acredolo (Ed.), *Spatial Orientation: Theory, Research, and Application*, pp. 225-282, Plenum Press, New York, NY.

S. Timpf, G. Volta, D. Pollock and M. Egenhofer (1992). A Conceptual Model of Wayfinding Using Multiple Levels of Abstraction. in: A. U. Frank, I. Campari, and U. Formentini (Eds.), *Theory and Methods of Spatial Temporal Reasoning in Geographic Space*, Pisa, Italy, Springer-Verlag, New York, NY, pp. 348-367.

N. Tryfona and M. Egenhofer (1997). Consistency Among Parts and Aggregates: A Computational Model. *Transactions in GIS* 1(3): 189-206.

A. Tversky (1977). Features of Similarity. *Psychological Review* 84(4): 327-352.

Z. Wang and J.-C. Muller (1993). Complex Coast Line Generalization. *Cartography and Geographic Information Systems* 20(2): 96-106.

R. Weibel (1995). Three Essential Building Blocks for Automated Generalization. in: J.-C. Muller, J.-P. Lagrange and R. Weibel (Eds.), *GIS and Generalization – Methodology and Practice*, pp. 56-69, Taylor & Francis, Bristol - PA.

R. Weibel (1996). A Typology of constraints to Line Simplification. in: M. J. Kraak and M. Molenaar (Eds.), *7th International Symposium on Spatial Data Handling*, Delft, The Netherlands, Taylor & Francis, pp. 9A.1-9A.14.

W. R. Weibel and J. S. DeLotto (1988). Automated Terrain Classification for GIS Modeling. in: *Proceedings LIS/GIS*, San Antonio, TX, pp. 618-627.

I. Zycor (1984). *Manual and automated line generalization and feature displacement*, Technical Report, US Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

Appendix: Classes Specification

A.1 Feature

Derived from SObject

Protected Attributes:

Fid : long

feature identifier.

Frepres : char

feature representation ('0'=undefined,'1'=point,'2'=linear,'3'=area).

Fboundary : BoundaryList

meet boundary components.

Fgeoid : long

geometry identifier.

Public Operations:

Feature () : Feature

Constructor.

~Feature () :

Destructor.

Id (i : long) void

Sets the feature identifier.

Input:

i: identifier.

Id () : long

Returns the feature identifier.

AppendMeetBoundary (b : Boundary*) : Boundary*

Adds a new boundary to feature list of boundaries, and returns this boundary pointer.

Input:

b: boundary pointer.

RemoveMeetBoundary () : Boundary*

Removes the current boundary from the list of boundaries and returns its pointer.

Representation () : char

Returns the feature representation type.

GeometryId () : long

Returns the geometric representation identifier.

GeometryId (id : long) : void

Sets the geometric representation identifier.

Input:

id: geometry identifier.

NumberOfBoundaries () : long

Returns the number of boundaries.

Boundaries () : BoundaryList*

Returns a pointer to the list of boundaries.

TotalMeets (meet0 : long&, meet1 : long&, mixmeet : long&) : void

Gets the number of meet types related with its adjacent elements.

Output:

meet0: number of 0-dimensional meets.

meet1: number of 1-dimensional meets.

mixmeet: number of mixed meets.

TotalBoundaries (meet0 : long&, meet1 : long&) : void

Gets the number of boundary types.

Output:

meet0: number of 0-dimensional boundaries.

meet1: number of 1-dimensional boundaries.

TotalMeetRelations () : long

Returns the number of meet relations.

AdjacentFeatures (flist : FeatureList*) : void

Gets the list of adjacent features.

Output:

flist: list of adjacent features.

NumberOfAdjacentFeatures () : long

Returns the number of adjacent features.

IsAdjacent (finput : Feature*) : short

Verifies if input feature is adjacent to this features.

Returns TRUE or FALSE.

Input:

finput: input feature.

Relation (finput : Feature*) : char

Returns the spatial relation with input feature.

Input:

finput: input feature.

Similar (finput : Feature*) : short

Returns TRUE if identifier and representation type are the same.

Input:

finput: input feature.

AdjacentStructure () : long

Returns how many meet relations exist between the adjacent features.

DimensionSimilarity (finput : Feature*) : double

Returns the dimension similarity measure compared with input feature.

Input:

finput: input feature.

MeetSimilarity (finput : Feature*) : double

Returns the number of meets similarity measure compared with input feature.

Input:

finput: input feature.

AdjacentSimilarity (flist : FeatureList*) : double

Returns the adjacent similarity measure using the adjacent elements of the input features.

Input:

flist: list of features.

HierarchySimilarity (finput : Feature*) : double

Returns the hierarchy similarity measure compared with input feature.

Input:

finput: input feature.

SimilarityMeasure (finput : Feature*) : double

Returns the total similarity measure compared with input feature.

Input:

finput: input feature.

A.2 PointFeature

Derived from Feature

Public Operations:

PointFeature () : PointFeature

Constructor.

~PointFeature () :

Destructor.

A.3 LinearFeature

Derived from Feature.

Public Operations:

LinearFeature () : LinearFeature

Constructor.

~LinearFeature () :

Destructor.

A.4 AreaFeature

Derived from Feature.

Private Attributes:

AFgraph : GraphList*

set of connected and isolated elements contained by this area feature.

AFboundary : BoundaryList

covers boundary components.

Public Operations:

AreaFeature () : AreaFeature

Constructor.

~AreaFeature () :

Destructor.

FirstGraph () : Graph*

Returns pointer to first graph at the lower level of the hierarchy (if it exists, otherwise returns NULL).

NextGraph () : Graph*

Returns pointer to next graph at the lower level of the hierarchy (if it exists, otherwise returns NULL).

AppendGraph (g : Graph*) : Graph*

Adds a new graph on the lower level hierarchy and returns the pointer to this input graph.

Input:

g: graph pointer.

Sons () : long

Returns the number of lower level graphs.

CoverBoundaries () : BoundaryList*

Returns a pointer to the list of cover boundaries.

AppendCoverBoundary (b : Boundary*) : Boundary*

Adds a new boundary on the list of cover boundaries and returns the

pointer to this input boundary.

Input:

b: boundary pointer.

NumberOfCoverBoundaries () : long

Returns the number of cover boundaries.

GeneralizedRelation (finput : Feature*) : char

Returns the spatial relation - cover type - with the input feature,
which must be in one of the lower level graphs under this area feature.

Input:

finput: pointer to lower level feature.

A.5 Boundary

Derived from SObject.

Private Attributes:

Bid : long

identifier.

Bfeatfirst : Feature*

first feature pointer.

Bsecfeat : Feature*

second feature pointer.

Btype : char

boundary type.

Bcomprel : char

complement relationship.

Bgeoids : SIdList

geometric representations of the boundary.

Public Operations:

Boundary () : Boundary

Constructor.

~Boundary () :

Destructor.

Define (id : long, first : Feature*, second : Feature*, type : char, comprel :
char, geom : long) : void

Defines the boundary attributes.

Input:

| | |
|----------|---------------------------------------|
| id: | identifier. |
| first: | pointer to first feature. |
| second: | pointer to second feature. |
| type: | boundary dimension and type. |
| cmprrel: | complement relationship. |
| geom: | first geometry identifier (may be 0). |

Id (id : long) void

Sets the boundary identifier.

Input:

| | |
|-----|-------------|
| id: | identifier. |
|-----|-------------|

Id () : long

Returns the boundary identifier.

FirstFeature (f : Feature*) : void

Sets the pointer to first feature.

Input:

| | |
|----|------------------------|
| f: | first feature pointer. |
|----|------------------------|

FirstFeature () : Feature*

Returns the pointer to first feature.

SecondFeature (f : Feature*) : void

Sets the pointer to second feature.

Input:

| | |
|----|-------------------------|
| f: | second feature pointer. |
|----|-------------------------|

SecondFeature () : Feature*

Returns the pointer to second feature.

Type (t : char) : void

Sets the type (contains the spatial relation and dimension type).

Input:

| | |
|----|----------------|
| t: | boundary type. |
| | ‘1’ = MEET0 |
| | ‘2’ = MEET1 |
| | ‘5’ = COVER0 |
| | ‘6’ = COVER1 |

Type () : char

Returns the boundary type.

ComplementRelationship (cr: char) : void

Sets the boundary complement relationship.

Input:

| | |
|-----|--------------------------|
| cr: | complement relationship. |
| | ‘1’ = BOUNDED |
| | ‘2’ = ‘UNBOUNDED |

ComplementRelationship () : char

Returns the complement relationship.

GeometryIds () : SIdList*

Returns a pointer to the list of geometry identifiers.

AddGeometryId (id : long) : short

Adds a new geometry that represents this boundary.

Input:

| | |
|-----|----------------------|
| id: | geometry identifier. |
|-----|----------------------|

A.6 Graph

Derived from SObject.

Private Attributes:

Gid : long

graph identifier.

Gfeature : FeatureList

list of features.

Gboundary : BoundaryList

list of boundaries.

Gfather : Graph*

higher level graph.

Gsuperior : Feature*

higher level area feature.

Public Operations:

Graph () : Graph

Constructor.

~Graph () :

Destructor.

Id (id : long) : void

Sets the graph identifier.

Input:

id: identifier.

Id () : long

Returns the graph identifier.

FirstFeature () : Feature*

Returns pointer to the first feature.

NextFeature () : Feature*

Returns pointer to the next feature.

AppendFeature (f : Feature*) : Feature*

Adds the input feature into graph, and returns the feature pointer if successful, otherwise returns NULL.

Input:

f: feature pointer.

NumberOfFeatures () : long

Returns the number of features.

Features () : FeatureList*

Returns pointer to feature list.

Boundaries () : BoundaryList*

Returns pointer to boundary list.

FirstBoundary () : Boundary*

Returns pointer to first boundary.

NextBoundary () : Boundary*

Returns pointer to next boundary.

AppendBoundary (b : Boundary*) : Boundary*

Adds the input boundary into graph, and returns the boundary pointer if successful, otherwise returns NULL.

NumberOfBoundaries () : long

Returns the number of boundaries.

FatherGraph (g : Graph*) : void

Sets the father graph from the hierarchical structure.

Input:

g: father graph pointer.

FatherGraph () : Graph*

Returns the father graph pointer.

FatherFeature (f : Feature*) : void

Sets the father feature from the hierarchical structure.

Input:

f: father feature pointer.

FatherFeature () : Feature*

Returns the father feature pointer.

GetFeature (id : long, type : char) : Feature*

Returns the pointer to the feature that matches the input parameters. If none found, returns NULL.

Input:

id: feature identifier.

type: feature representation type.

RemoveFeature (id : long, type : char) : Feature*

Removes from the graph the feature that matches the input parameters, and returns the pointer to this feature.

Input:

id: feature identifier.

type: feature representation type.

GetBoundary (id : long) : Boundary*

Returns the boundary pointer that matches the input identifier.

Input:

id: boundary identifier.

RemoveBoundary (id : long) : Boundary*

Removes from the graph the boundary that matches the input parameter, and returns the pointer to this boundary.

Input:

id: boundary identifier.

FeatureDimension (point : long&, line : long&, region : long&) : void

Counts the number of each feature dimensions present on graph.

Output:

point: number of point features.

line: number of linear features.

region: number of area features.

Relations (meet0 : long&, meet1 : long&, mixmeet : long&, disjoint : long&) : void

Counts the number of spatial relations present on graph.

Output:

meet0: number of 0-dimensional meets.

meet1: number of 1-dimensional meets.

mixmeet: number of mixed meets.

disjoint: number of disjoint relations.

GeneralizedRelations (cov0 : long&, cov1 : long&, mixcov : long&, contain : long&, equal : long&) : void

Counts the number of spatial relations related with father graph.

Output:

cov0: number of 0-dimensional covers.

cov1: number of 1-dimensional covers.

mixcov: number of mixed covers.

contain: number of contain relations.

equal: number of equal relations.

BuildBoundaries (pset : PolygonSet&, ndset : NodeSet&, next : long&) : short

Builds the graph boundary from SPRING vector model.

Input:

pset: reference to SPRING polygon set.

ndset: reference to SPRING node set.

HasSubGraphs () : short

Returns if exist lower level graphs.

HierarchyLevel () : short

Returns the level of this graph on the general hierarchy.

GetNumberOfBoundaries (zdbound : long&, odbound : long&) : void

Counts the number of meet boundaries per type.

Input:

zdbound: number of 0-dimensional meets.

odbound: number of 1-dimensional meets.

A.7 SpatialScene

Private Attributes:

SSnextsetid : long

next available graph identifier.

SSnextfeatid : long

next available feature identifier.

SSnextboundid : long

next available boundary identifier.

SSsets : GraphList

list of graphs.

Public Operations:

SpatialScene () : SpatialScene

Constructor.

~SpatialScene () :

Destructor.

AppendGraph (g : Graph*) : Graph*

Adds a new graph to spatial scene, and returns its pointer.

Input:

g: graph pointer.

FirstGraph() : Graph*

Returns the first graph pointer for this scene.

NextGraph () : Graph*

Returns the next graph pointer for this scene.

GetGraph (id : long) : Graph*

Returns the graph pointer from input identifier.

Input:

id: graph identifier.

GetGraph (id : long, type : char) : Graph*

Returns the pointer to the graph that contains the feature specified by the input parameters.

Input:

id: feature identifier.

type: feature representation type.

Graphs () : GraphList*

Returns the pointer to the list of graphs of this scene.

NumberOfGraphs () : long

Returns the number of graphs in this scene.

Read (filename : char*) : short

Reads the scene contents from an ascii file.

Input:

filename: disk file name.

Write (filename : char*) : short

Write the scene contents into an ascii file.

Input:

filename: disk file name.

GetFeature (id : long, type : char) : Feature*

Returns the feature pointer that has the input parameters.

Input:

id: feature identifier.

type: feature representation type.

Build (lset : LineSet&, ndset : NodeSet&, pset : PolygonSet&) : short

Transforms the SPRING model into the relation-based model.

Input:

lset: line set of SPRING vector model.

ndset: node set of SPRING vector model.

pset: polygon set of SPRING vector model.

Print () : void

Prints the scene contents into the standard output device.

Clear () : void

Clears the scene contents.

Merge (first : AreaFeature&, second : AreaFeature&) : AreaFeature*

Performs a merge operation between two area features,
and returns the merged feature.

Input:

first: area feature reference.

second: area feature reference.

Drop (feat : Feature&) : short

Drops input feature from scene.

Input:

feat: feature reference.

NumberOfLevels () : short

Returns the number of levels of this scene.

NumberOfGraphsAtLevel (level : short) : short

Returns the number of graphs at a specified level of hierarchy.

Input:

level: hierarchy level.

GetGraphsAtLevel (level : short, glist : GraphList&) : short

Gets the graphs at a specified level.

Input:

level: hierarchy level.

Output:

glist: list of graphs.

GetNumberOfBoundaries (zdbound : long&, odbound : long&) : void

Gets the number of boundary components.

Output:

zdbound: number of 0-dimensional boundaries.

odbound: number of 1-dimensional boundaries.

LevelSimilarity (input : SpatialScene*) : double

Returns the similarity measure based on the number of hierarchical levels.

Input:

input: scene to be compared.

GraphSimilarity (input : SpatialScene*) : double

Returns the similarity measure based on the number of graphs per level of hierarchy.

Input:

input: scene to be compared.

EvaluateTopology (input : SpatialScene*, miso : MultiIsomorphicConfList&) : short

Evaluates the topological equivalence between scens.

Input:

input: scene to be compared.

Output:

miso: collection of isomorphic configurations.

Private Operations:

BuildBoundaries (pset : PolygonSet&, ndset : NodeSet&) : short

Generates the boundary components on the relation-based model, using the polygon and node sets of SPRING vector model.

Input:

pset: reference to SPRING polygon set.

ndset: reference to SPRING node set.

BuildFeatures (pset : PolygonSet&, ndset : NodeSet&) : short

Generates the graph nodes (features) from SPRING vector model.

Input:

pset: reference to SPRING polygon set.

ndset: reference to SPRING node set.

BuildGraphs (gfather : Graph*, ffather : Feature*, plist : SPolygonList&, ndset : NodeSet&) : short

Generates the scene graph representation.

Input:

gfather: pointer to father graph.

ffather: pointer to father feature.

plist: SPRING polygon list.

ndset: SPRING node set.

AddAdjacentElements (g : Graph&, poly : SPolygon&, plist : SPolygonList&, ndset : NodeSet&) : short

Adds adjacent elements of input polygon into input graph.

Input:

g: reference to current graph.

poly: reference to input SPRING polygon.

plist: SPRING polygon list.
ndset: SPRING node set.

A.8 MatchingPair

Derived from SObject.

Private Attributes:

MPpair : PairOfFeature

pair of features.

MPvalid : short

identifies if pair is a valid match.

Public Operations:

MatchingPair () : MatchingPair

Constructor.

~MatchingPair () :

Destructor.

PairOfFeature (p : PairOfFeature*) : void

Defines the pair of features.

Input:

p: pair of features pointer.

PairOfFeature () : PairOfFeature*

Returns pointer to pair of features.

FirstFeature () : Feature*

Returns pointer to first feature.

SecondFeature () : Feature*

Returns pointer to second feature.

FirstFeatureId () : long

Returns the first feature identifier.

FirstFeatureType () : char

Returns the first feature representation type.

SecondFeatureId () : long

Returns the second feature identifier.

SecondFeatureType () : char

Returns the second feature representation type.

IsValid (v : short) : void

Sets if pair of features is a valid association considering the boundary sequence.

Input:

v: valid flag - TRUE or FALSE.

IsValid () : short

Returns if pair of features is a valid association considering the boundary sequence.

A.9 IsomorphicConf

Derived from SObject, MatchingPairList.

Private Attributes:

ICfirst : Graph*

pointer to first feature.

ICsecond : Graph*

pointer to second feature.

Public Operations:

IsomorphicConf () : IsomorphicConf

Constructor.

~IsomorphicConf () :

Destructor.

IsValid () : short

Returns (TRUE or FALSE) if this configuration contains all valid pair of features.

SetGraphs (first : Graph*, second : Graph*) : void

Defines the graph pointers.

Input:

first: first graph pointer.

second: second graph pointer.

FirstGraph () : Graph*

Returns pointer to first graph.

SecondGraph () : Graph*

Returns pointer to second graph.

IsPresent (pflist : PairOfFeatureList&) : short

Verifies if the input pair of features list is part of this isomorphic configuration. Returns TRUE or FALSE.

Input:

pflist: reference to list of pair of features.

IsPresent (pair : PairOfFeature&) : short

Verifies if the input pair of features is part of this isomorphic configuration. Returns TRUE or FALSE.

Input:

pair: reference to pair of feature.

ValidSimilarity () : double

Calculates the valid similarity measure. It counts how many valid pair of features are, and divide by the size of this configuration.

TopologicalSimilarity () : double

Calculates the topological similarity measure.

DimensionSimilarity () : double

Calculates the dimension similarity measure.

EvaluateBoundarySequence () : short

Evaluates the boundary sequence components of each pair of features, and returns TRUE (if they are ok) or FALSE.

GetPair (pair : PairOfFeature&) : MatchingPair*

Returns the matching pair associated with input pair of feature.

Input:

pair: reference to pair of features.

PairWithFirst (pair : Feature&) : MatchingPair*

Returns the pointer to the pair of features that contains as a first feature the input parameter.

Input:

pair: reference to pair of features.

HasAsFirstFeature (id : long, type : char) : Feature*

Returns a pointer to a feature, if the configuration has one pair in which the first feature is equivalent with input parameters.

Input:

id: feature identifier.

type: feature representation type.

HasAsSecondFeature (id : long, type : char) : Feature*

Returns a pointer to a feature, if the configuration has one pair in which the first feature is equivalent with input parameters.

Input:

id: feature identifier.

type: feature representation type.

HierarchyLevel () : short

Returns the hierarchical level of this configuration.

A.10 AssociationGraph

Private Attributes:

AGfirst : Graph*

pointer to first feature.

AGsecond : Graph*

pointer to second feature.

AGnodes : PairOfFeatureList

list of pair of features.

Public Operations:

AssociationGraph () : AssociationGraph

Constructor.

~AssociationGraph () :

Destructor.

DefineGraphs (first : Graph*, second : Graph*) : void

Sets the graph pointers.

Input:

first: first graph pointer.

second: second graph pointer.

Build (dimdist : short = 0, sptreldist : short = 0, cinvdist : short = 0) : short

Builds the association pair of features between the graphs.

Input:

dimdist: dimension distance relaxation.

sptreldist: spatial relations distance relaxation.

cinvdist: component invariant distance relaxation.

FindIsomorphism (isolist : IsomorphicConfList&) : short

Finds the isomorphic configurations based on the current pair association.

Output:

isolist: reference to list of isomorphic configurations.

NumberOfNodes () : long

Returns the number of nodes in this association graph.

FirstNode () : PairOfFeature*

Returns the pointer to the pair of features that corresponds to the first node of this association graph.

NextNode () : PairOfFeature*

Returns the pointer to the pair of features that corresponds to the next node of this association graph.

AppendNode (pair : PairOfFeature*) : PairOfFeature*

Adds a new node on this association graph using the input pair

of features.

Input:

pair: pair of features pointer.

Private Operations:

CheckGraph (pair : PairOfFeature&, iso : IsomorphicConf&) : short

Recursively check the equivalence between features and builds the isomorphic configurations.

Input:

pair: reference to current pair of features.

Output:

iso: isomorphic configuration

Biography of the Author

João A. C. Paiva was born in Resende - RJ - Brazil on January 6, 1963. He attended the Federal University of Espírito Santo and graduated in 1985 with a degree in Electrical Engineering. In 1985 he started to work at the National Institute for Space Research (INPE) in São José dos Campos - SP at the Image Processing Division developing software for geographic applications. He enrolled in a Master's program for Computer Science at INPE and completed this course in 1994. One year later he got a leave of absence from INPE and entered the Graduate program of the Spatial Information and Science Engineering at University of Maine, concentrating his studies on topological aspects of geographic databases with multiple representations.

After receiving his degree, João will be returning to Brazil to resume his position as a Senior Engineer at INPE. He is a candidate for the Doctor of Philosophy degree in Spatial Information Science and Engineering from the University of Maine in December, 1998.